



THE BUILDERS PROJECT'S

GUIDE TO BUILDING

THE AURORA TOOLSET MANUAL

VERSION 1.06



TABLE OF CONTENTS

CHAPTER 1	USING THE TOOLSET.....	11
1.1	THE TOOLSET WINDOWS	11
1.1.1	<i>The Module Structure Window.....</i>	<i>11</i>
1.1.2	<i>The Main View Window</i>	<i>12</i>
1.1.3	<i>The Palette.....</i>	<i>13</i>
CHAPTER 2	MODULES.....	16
2.1	CREATING A NEW MODULE	16
2.2	THE PROPERTIES OF A MODULE.....	20
2.2.1	<i>The Basic Properties of a Module.....</i>	<i>20</i>
2.2.2	<i>The Events Properties of a Module.....</i>	<i>22</i>
2.2.3	<i>The Advanced Properties of a Module.....</i>	<i>26</i>
2.2.4	<i>The Description of a Module.....</i>	<i>29</i>
2.2.5	<i>The Custom Content of a Module.....</i>	<i>29</i>
2.2.6	<i>The Cached Scripts of a Module</i>	<i>30</i>
2.3	DESIGNING TIPS	32
2.4	HOW TO... ..	32
2.4.1	<i>...have multiple starting locations in your module.....</i>	<i>32</i>
2.4.2	<i>...have the player characters respawning after death in a specific location.</i>	<i>33</i>
CHAPTER 3	AREAS.....	34
3.1	CREATING AN AREA WITH THE AREA WIZARD.....	34
3.2	THE PROPERTIES OF AN AREA.....	34
3.2.1	<i>The Basic Area Properties</i>	<i>36</i>
3.2.2	<i>The Visual Area Properties.....</i>	<i>37</i>
3.2.3	<i>The Audio Area Properties.....</i>	<i>41</i>
3.2.4	<i>The Events Area Properties.....</i>	<i>43</i>
3.2.5	<i>The Advanced Area Properties</i>	<i>45</i>
3.2.6	<i>The Comments Area Properties</i>	<i>46</i>
3.3	DESIGNING TIPS	47
3.4	HOW TO... ..	47

3.4.1	<i>...copy an area</i>	47
3.4.2	<i>... modify the properties of a specific tile</i>	48
3.4.3	<i>...reveal the map of an area to a player</i>	48
CHAPTER 4	CREATURES	50
4.1	CREATING A CREATURE BLUEPRINT WITH THE CREATURE WIZARD	50
4.2	CREATING A CREATURE BLUEPRINT FROM AN EXISTING CREATURE	55
4.3	THE PROPERTIES OF A CREATURE	56
4.3.1	<i>The Basic Properties of a Creature</i>	58
4.3.2	<i>The Statistics Properties of a Creature</i>	61
4.3.3	<i>The Appearance Properties of a Creature</i>	62
4.3.4	<i>The Classes Properties of a Creature</i>	63
4.3.5	<i>The Skills Properties of a Creature</i>	65
4.3.6	<i>The Scripts Properties of a Creature</i>	66
4.3.7	<i>The Advanced Properties of a Creature</i>	69
4.3.8	<i>The Feats Properties of a Creature</i>	73
4.3.9	<i>The Spells Properties of a Creature</i>	74
4.3.10	<i>The Special Abilities Properties of a Creature</i>	76
4.3.11	<i>The Comments Properties of a Creature</i>	77
4.3.12	<i>The Inventory</i>	77
4.4	THE FACTIONS	79
4.5	DESIGNING TIPS	84
4.6	HOW TO	85
4.6.1	<i>...add animations to my NPCs to make them more alive</i>	85
4.6.2	<i>...make a dead creature</i>	85
4.6.3	<i>...make my creatures drop crafting items</i>	86
CHAPTER 5	CONVERSATIONS	87
5.1	CREATING A CONVERSATION FILE	87
5.1.1	<i>Writing your conversation</i>	87
5.1.2	<i>Integrating your conversation in the toolset</i>	87
5.2	THE CONVERSATION EDITOR	88
5.3	THE STRUCTURE OF A CONVERSATION	89
5.3.1	<i>Creating your conversation tree</i>	89
5.3.2	<i>Using links</i>	90
5.3.3	<i>Adding conditional nodes</i>	92
5.3.4	<i>Creating a multi-party conversation</i>	95
5.3.5	<i>Inserting tokens and highlighting text</i>	96

5.3.6	<i>Creating conversations in multiple languages</i>	98
5.4	THE PROPERTIES OF A CONVERSATION NODE	98
5.4.1	<i>The “Text Appears When...” Property</i>	98
5.4.2	<i>The “Action Taken” Property</i>	98
5.4.3	<i>The “Other Actions” Property</i>	100
5.4.4	<i>The Comments Property</i>	101
5.4.5	<i>The “Current File” Property of the Conversation File</i>	101
5.5	OTHER CONVERSATION FUNCTIONS	102
5.5.1	<i>Testing a Conversation</i>	102
5.5.2	<i>Bookmarks</i>	102
5.5.3	<i>Conversation Editor’s Options</i>	103
5.6	RUNNING A CONVERSATION	104
5.7	DESIGNING TIPS	105
5.7.1	<i>Starting Conditional Scripts</i>	106
5.8	HOW TO.....	108
5.8.1	<i>... have a NPC remember a player from a previous conversation</i>	108
5.8.2	<i>... have a henchman intervening in a conversation</i>	109
CHAPTER 6	JOURNAL	110
6.1	JOURNAL CATEGORIES	110
6.2	JOURNAL ENTRIES	111
6.3	UPDATING THE JOURNAL FOR PLAYERS.....	112
6.3.1	<i>Updating the Journal Through a Conversation</i>	113
6.3.2	<i>Updating the Journal Through a Script</i>	113
6.4	DESIGNING TIPS	113
CHAPTER 7	ITEMS	114
7.1	CREATING AN ITEM BLUEPRINT WITH THE ITEM WIZARD	114
7.2	CREATING AN ITEM BLUEPRINT FROM AN EXISTING BLUEPRINT	117
7.3	THE PROPERTIES OF AN ITEM	117
7.3.1	<i>The General Item Properties</i>	118
7.3.2	<i>The Appearance of an item</i>	122
7.3.3	<i>The Magical Item Properties</i>	122
7.3.4	<i>The Description of an Item</i>	124
7.3.5	<i>The Comments Property of an Item</i>	125
7.4	DESIGNING TIPS	125
7.5	HOW TO.....	126
7.5.1	<i>...implement or remove the item level restriction rule on my module</i>	126

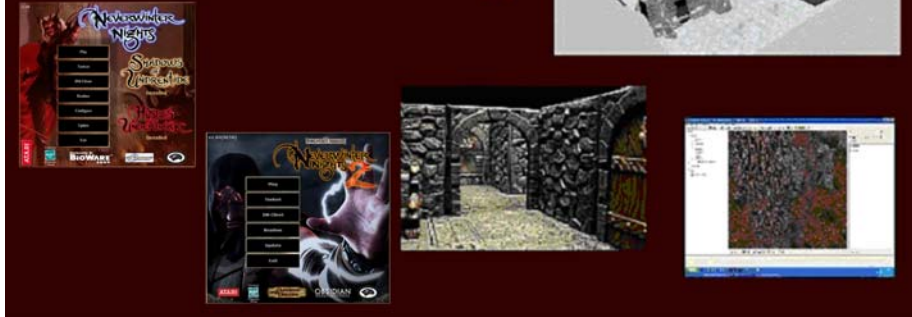
CHAPTER 8	PLACEABLES.....	127
8.1	CREATING A PLACEABLE BLUEPRINT WITH THE PLACEABLE WIZARD	127
8.2	CREATING A PLACEABLE BLUEPRINT FROM AN EXISTING ITEM	128
8.3	THE PROPERTIES OF A PLACEABLE OBJECT	128
8.3.1	<i>The Basic Properties of a Placeable Object.....</i>	<i>128</i>
8.3.2	<i>The Lock of a Placeable Object</i>	<i>130</i>
8.3.3	<i>The Trap of a Placeable Object</i>	<i>131</i>
8.3.4	<i>The Scripts of a Placeable Object.....</i>	<i>133</i>
8.3.5	<i>The Advanced Properties of a Placeable Object.....</i>	<i>135</i>
8.3.6	<i>The Description of a Placeable Object.....</i>	<i>137</i>
8.3.7	<i>The Comments on a Placeable Object.....</i>	<i>137</i>
8.4	DESIGNING TIPS	138
8.5	HOW TO.....	138
8.5.1	<i>...make a placeable object engage in a conversation</i>	<i>138</i>
8.5.2	<i>...make my placeable drop crafting items when destroyed</i>	<i>138</i>
8.5.3	<i>...adjust the position of a placeable object.....</i>	<i>139</i>
CHAPTER 9	DOORS	141
9.1	THE PROPERTIES OF A DOOR.....	141
9.1.1	<i>The Basic Properties of a Door.....</i>	<i>142</i>
9.1.2	<i>The Properties of the Door Lock.....</i>	<i>143</i>
9.1.3	<i>The Trap on a Door.....</i>	<i>144</i>
9.1.4	<i>The Area Transition Parameters of a Door.....</i>	<i>145</i>
9.1.5	<i>The Script Events of a Door.....</i>	<i>147</i>
9.1.6	<i>The Advanced Properties of a Door.....</i>	<i>148</i>
9.1.7	<i>The Description of a Door.....</i>	<i>150</i>
9.1.8	<i>The Comments on a Door.....</i>	<i>150</i>
9.2	DESIGNING TIPS	150
9.3	HOW TO.....	151
9.3.1	<i>...make my doors close themselves after a certain delay.....</i>	<i>151</i>
9.3.2	<i>...Make my doors drop crafting items when destroyed</i>	<i>152</i>
9.3.3	<i>...make free standing doors.....</i>	<i>152</i>
CHAPTER 10	ENCOUNTERS	153
10.1	CREATING AN ENCOUNTER BLUEPRINT WITH THE ENCOUNTER WIZARD	153
10.2	THE PROPERTIES OF AN ENCOUNTER	155
10.2.1	<i>The Basic Properties of an Encounter.....</i>	<i>155</i>
10.2.2	<i>The Creature List of an Encounter.....</i>	<i>157</i>

10.2.3	<i>The Scripts related to an Encounter</i>	157
10.2.4	<i>The Advanced Properties of an Encounter</i>	159
10.2.5	<i>The Comments on an Encounter</i>	160
10.3	SPAWNING POINTS.....	160
10.4	DESIGNING TIPS.....	161
CHAPTER 11	SOUNDS	163
11.1	CREATING A SOUND BLUEPRINT WITH THE SOUND WIZARD.....	163
11.2	THE PROPERTIES OF A SOUND OBJECT.....	166
11.2.1	<i>The Basic Properties of a Sound Object</i>	166
11.2.2	<i>The Positioning of a Sound Object</i>	169
11.2.3	<i>The Advanced Properties of a Sound Object</i>	171
11.3	DESIGNING TIPS.....	172
11.4	HOW TO.....	173
11.4.1	<i>...make a creature play a sound</i>	173
CHAPTER 12	TRIGGERS	174
12.1	ADDING A TRIGGER.....	174
12.2	CATEGORIES OF TRIGGER.....	175
12.3	CREATING A NEW TRIGGER BLUEPRINT.....	175
12.4	THE PROPERTIES OF A TRIGGER OBJECT.....	175
12.4.1	<i>The Basic Properties of a Trigger Object</i>	176
12.4.2	<i>The Script Events of a Trigger Object</i>	176
12.4.3	<i>The Advanced Properties of a Trigger Object</i>	177
12.4.4	<i>The Comments on a Trigger Object</i>	179
12.5	PROPERTIES SPECIFIC TO AREA TRANSITIONS TRIGGERS.....	179
12.6	PROPERTIES SPECIFIC TO TRAP TRIGGERS.....	180
12.7	BUILT-IN GENERIC TRIGGERS.....	180
12.7.1	<i>Henchman Trigger</i>	180
12.7.2	<i>Tracks Trigger</i>	180
12.7.3	<i>X0_SAFEREST Trigger</i>	181
12.7.4	<i>XP2 Interjection/One-Liner, Non-Random/Random Triggers</i>	181
12.8	DESIGNING TIPS.....	181
12.9	HOW TO.....	182
12.9.1	<i>...make the Secret Object Triggers work</i>	182
12.9.2	<i>...make a custom placeable object appear while using a Secret Object Trigger</i>	182
12.9.3	<i>...have an area transition trigger appear in game even though no actual area transition event has been assigned to it</i>	183
CHAPTER 13	WAYPOINTS	184

13.1	CREATING A NEW WAYPOINT BLUEPRINT.....	184
13.2	THE PROPERTIES OF A WAYPOINT	184
13.2.1	<i>The Basic Properties of a Waypoint</i>	<i>185</i>
13.2.2	<i>The Advanced Properties of a Waypoint.....</i>	<i>185</i>
13.3	BUILT-IN WAYPOINT BLUEPRINTS.....	186
13.4	DESIGNING TIPS	186
13.5	HOW TO.....	186
13.5.1	<i>...have a creature walk amongst waypoints</i>	<i>186</i>
13.5.2	<i>...make walking routines cross an area transitions</i>	<i>187</i>
CHAPTER 14	MERCHANTS	188
14.1	CREATING A NEW MERCHANT BLUEPRINT.....	188
14.2	HOW TO TURN A CREATURE INTO A STOREKEEPER.....	188
14.3	THE PROPERTIES OF A MERCHANT OBJECT	191
14.3.1	<i>The Basic Properties of a Merchant Object</i>	<i>191</i>
14.3.2	<i>The Advanced Properties of a Merchant Object</i>	<i>193</i>
14.3.3	<i>The Restrictions of a Merchant Object</i>	<i>195</i>
14.3.4	<i>The Comments on a Merchant Object.....</i>	<i>195</i>
14.4	DESIGNING TIPS	195
CHAPTER 15	HENCHMEN	196
15.1	HOW TO TURN A CREATURE INTO A HENCHMAN.....	196
15.2	THE CONVERSATION FILE OF A HENCHMAN.....	196
15.2.1	<i>Conversation node spoken when the Henchman has not yet been hired</i>	<i>197</i>
15.2.2	<i>Conversation node spoken between a hired Henchman and his Master.....</i>	<i>197</i>
15.2.3	<i>Conversation node spoken between a hired Henchman and a player that is not his Master.....</i>	<i>201</i>
15.3	DESIGNING TIPS	201
15.4	HOW TO.....	202
15.4.1	<i>...allow players to have more than one henchman.....</i>	<i>202</i>
15.4.2	<i>...to bring a henchman back to life only if a resurrection / raise dead spell is cast on him/her</i>	<i>202</i>
CHAPTER 16	X2_INC_SWITCHES	203
16.1	INTERFACE FOR SWITCHING SUBSYSTEM FUNCTIONALITY	203
16.1.1	<i>Module Switches.....</i>	<i>204</i>
16.1.2	<i>Miscellaneous</i>	<i>205</i>
16.1.3	<i>Creatures</i>	<i>206</i>
16.1.4	<i>Doors</i>	<i>207</i>
16.1.5	<i>Waypoints</i>	<i>207</i>
16.1.6	<i>Items.....</i>	<i>207</i>

16.1.7	<i>Spellscripts</i>	207
16.2	DOING IT IN HEX CODE.....	208
16.2.1	<i>For those who are clueless like I was</i>	211
CHAPTER 17	XP2 (HOTU) REST / WANDERING MONSTER SYSTEM	212
17.1	THE BASICS.....	212
17.2	MAKING YOUR OWN WANDERING MONSTER TABLE (.2DA)	213
CHAPTER 18	USING THE SOU TREASURE SYSTEM	215
18.1	MAIN CONCEPT:	215
18.2	CONTAINER SETUP:	215
18.3	MONSTER/NPC TREASURE	216
18.3.1	<i>Tag based Monster Treasure</i>	216
18.3.2	<i>Racial Type Based Monster Treasure</i>	216
18.3.3	<i>Generic Monster Treasure</i>	217
CHAPTER 19	OTHER STUFF	219
19.1	ADDING CRAFTING MATERIALS TO YOUR CAMPAIGN.	219
19.2	USING HAKS TO PATCH YOUR MODULE AFTER RELEASE.	219
19.2.1	<i>The Purpose:</i>	219
19.2.2	<i>The Concept:</i>	219
19.2.3	<i>Preparation:</i>	220
19.2.4	<i>Patching Procedure:</i>	220
19.2.5	<i>Considerations:</i>	220
19.2.6	<i>Module Problem Example:</i>	220
19.2.7	<i>The Patch:</i>	220
19.2.8	<i>The Patch at Work:</i>	221
19.2.9	<i>Later Patches:</i>	221
19.2.10	<i>Final Words:</i>	221
19.3	ALTERNATE METHOD FOR DELETING BLUEPRINTS FROM YOUR MODULE	222
19.4	ALTERNATE METHOD FOR DELETING SCRIPTS FROM YOUR MODULE.....	222
CHAPTER 20	COMMON ACRONYMS	223
20.1.1	<i>General Online Communications</i>	223
20.1.2	<i>Gaming Specific Terms</i>	224
CHAPTER 21	RESOURCES	227
CHAPTER 22	CREDITS	230

The Builders Project's Guide To Building



INTRODUCTION

Neverwinter Nights, developed by BioWare and distributed by Atari (formerly Infogames), has become one of the leaders in end user adaptability. The main engine behind this is the Aurora Toolset, which allows users to create their own modules. No longer are they limited to what was developed by BioWare originally.

What we can do with the toolset is mind-boggling. However, with so much that you can do, where on earth do you start? That is where The Builders Project comes in. The Builders Project is a rapidly expanding group of Neverwinter Nights and/or Dungeons and Dragons enthusiasts who have dedicated the guild to a few seemingly simple objectives:

- i) To reach as many new module builders as possible and get them educated in the use of the NWN Toolset. We want to do this within the framework of the membership, but we see no reason to make any info published by the guild private in any way.
- ii) To provide a support system for module builders.
- iii) To gather, pool and make available any resources we encounter regarding module building, with the intent of aiding our members in the pursuit of module building.
- iv) To provide a means of sharing knowledge and experience in module building, foster an atmosphere of collective education, and further the purpose of making our membership competent and capable builders.

To that end, the Builders Project's editors have started collecting information on every aspect of module building and keeping track of the questions frequently asked on the guild's forum and the answers provided to these questions. All this information is stored and edited in what we call The Guide to Building.

The Guide to Building will be released in many instalments, each covering an important aspect of module building. This first instalment, titled "The Toolset Manual" looks at the toolset, the wizards and options contained therein. It is designed to basically cover every possibility, big or small, offered by the Toolset without the need to get into heavy scripting.

The Toolset Manual of the Guide to Building is:

- A reference manual in which every object type, property, tool and functionality offered by the Toolset is described in detail.
- A gathering of designing tips for module builders.
- A collection of frequently asked questions regarding the toolset.

The Toolset Manual of the Guide to Building is NOT:

- A tutorial on how to create a module.
- A scripting tutorial.
- A guide to provide you with information on how to create custom content (i.e. hakpacks).

Please note that this version of the Toolset Manual has been written while NWN and the SoJ and HotU expansions were in version 1.68.

Any help or support related to this instalment of the Guide to Building or any of our other guides may be obtained in our guild's Help Desk forum at the following address:

[The Builders Project Help Desk](#)

The Builders Project's Guild

CHAPTER 1 Using the Toolset



Does this screen look familiar? Well this is the gate to the toolset, a place where endless hours can be squandered, but not in vain. You will be amazed at all the possibilities this powerful tool has to offer and how well designed it is for builders.

In this chapter, we will simply give you an overview of how the toolset is presented.

As you may expect, in order to launch the toolset you must click on the “Toolset” button of the NWN launching screen. You will then be asked if you want to open an existing module or create a new one.

The creation of a new module is covered in [Chapter 2](#). If you do not have an existing module, you will need to create one first before you can gain access to the toolset windows. So if you are unfamiliar with the new module creation process, you might want to go through [Section 2.1 Creating a New Module](#) of this manual first in order to create your first module.

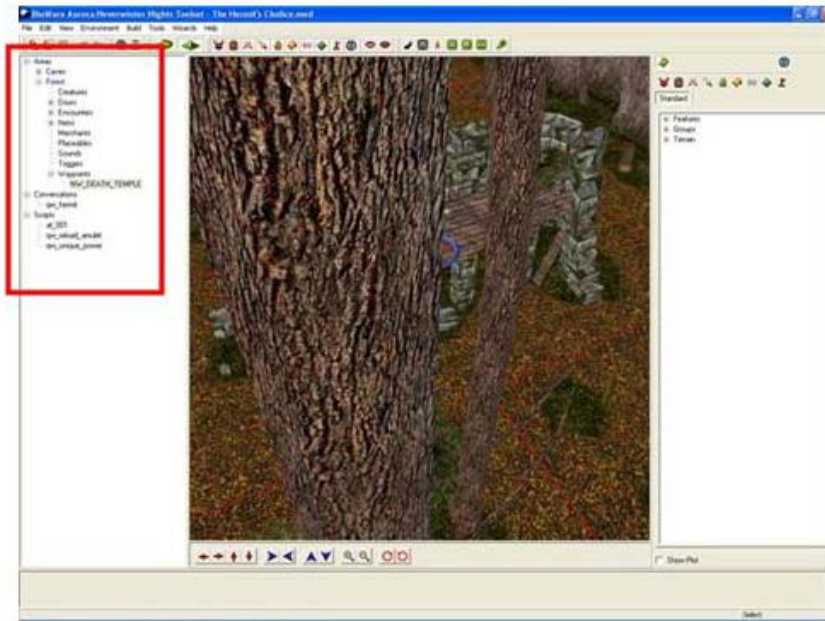
1.1 The Toolset Windows

There are three main parts to the toolset window: on the left is the module structure, in the center is the main view and on the right is the palette.

1.1.1 The Module Structure Window

In this window you will find a list of the resources that are present in your module. Every time you create a new area, place a new creature, or write a new conversation, it will show up somewhere in this window.

Image 1.1.1a – The Module Structure Window



There are three main roots to your module structure tree: Areas, Conversations and Scripts. The little boxes with either a “+” or “-” sign allow the lists of your module structure tree to be expanded or compressed.

By expanding the “Areas” root, you will get a list of every area that is included in your module. You can then expand the tree again for each specific area and you will get the opportunity to see the list of all creatures, doors, encounters, items, merchants, placeables, sounds, triggers and waypoints that were placed in that area. All these categories of objects have their own chapters in this manual, providing you with all the information you require to understand their use and how to customize them to fit your needs.

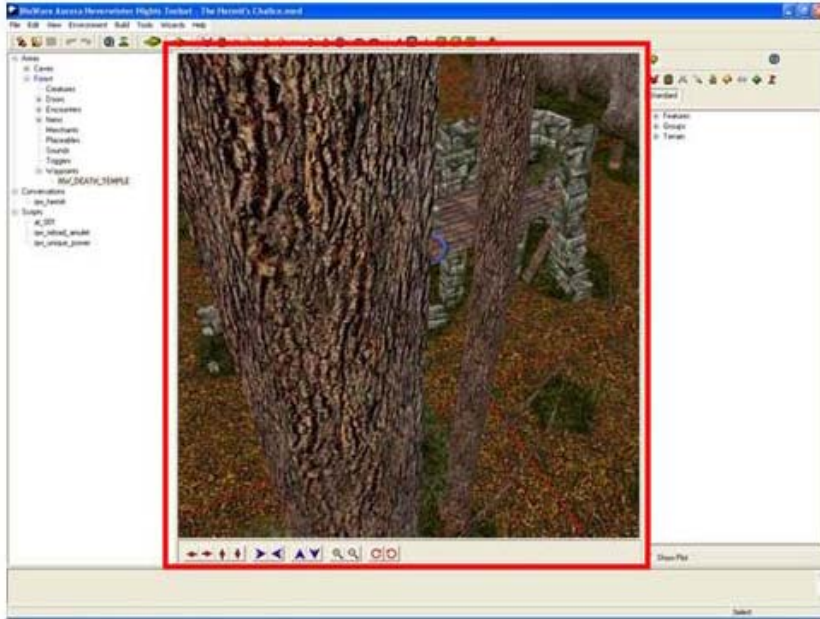
Remember that you can at any time double-click on any object or area listed in the module structure tree and the toolset will bring you, in the main view window, to this object or area.

Expanding the “Conversations” or “Scripts” roots will provide you with the list of conversation or script files that you have written in your module. Writing conversations is covered in [Chapter 5](#) of this manual.

1.1.2 The Main View Window

The main view window is the camera to the world you are building. It is in this window that most of the actual “building” gets done. As will be mentioned in the sub-section below presenting the palette window, it is in the main view window that you draw your areas and that you place the creatures and other objects that will be filling your world.

Image 1.1.2a – The Main View Window



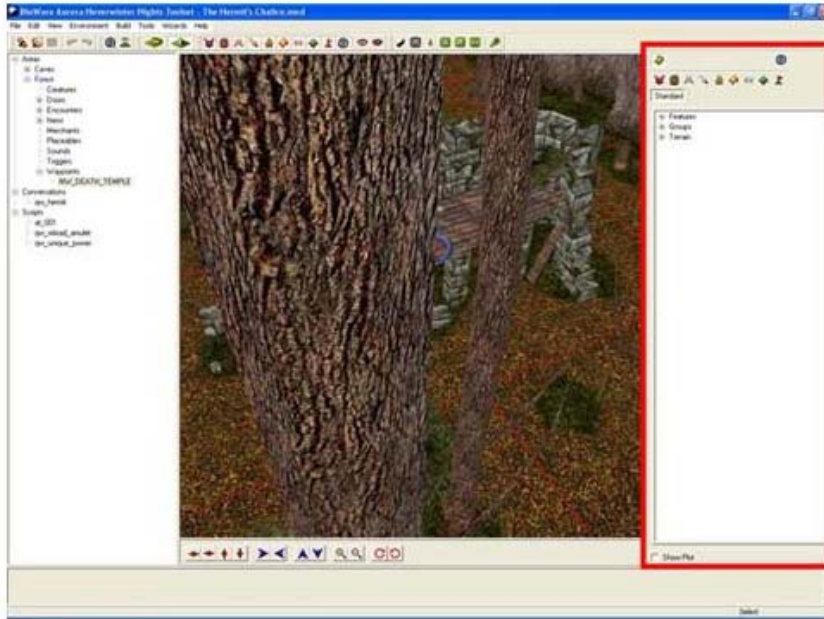
The buttons at the bottom of the main view window allow you to zoom in or out, and to turn or rotate either your camera or an object that is currently selected.

The buttons at the top of the main view window determine what types of objects you actually want to appear in the window. As an example, if you only want to see the creatures in an area and do not want to see every other object that might be obstructing your view, then you can disable every button but the creature one.

1.1.3 The Palette

The palette is basically your source for building up your world. In the palette you will find templates (which are called “Blueprints” in the toolset) for your creatures, doors, buildings, encounters, sounds, etc. Essentially everything you need to create your world is in the palette. You simply have to choose what fits your tastes and place it in the area that is currently opened in the toolset.

Image 1.1.3a – The Palette



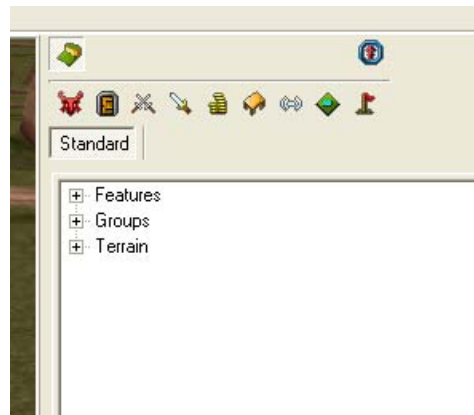
The Palette is composed of two categories of models: (1) tiles; and (2) objects that can be placed.

TILES:

Image 1.1.3b – The Palette – Tiles

Tiles are used to shape your area. The “Features” tiles are the land type tiles that only occupy one tile space (i.e. one square in your area). The “Groups” tiles are composed of larger land type tiles. The “Terrain” tiles are used to craft the ground of your areas.

The tiles that will be available for each area depend on the tileset you have used when you created the area that is opened in the toolset (see Section 3.1). The only way for you to use custom tiles or tilesets is by adding a hakpack to your module (see Sub-Section 2.2.5).



To add or modify features, groups or terrain in your area, select the chosen tile in the palette and then click in the Main View Window where you want the tile to be added. If your cursor in the Main View Window is red, it means that the terrain that is in place does not allow you to place this tile where the cursor is.

To rotate a Feature or Group tile that you have selected before you have placed it, you right-click with your mouse.

Note that the size of a tile is scaled to represent a 10 meters by 10 meters square.

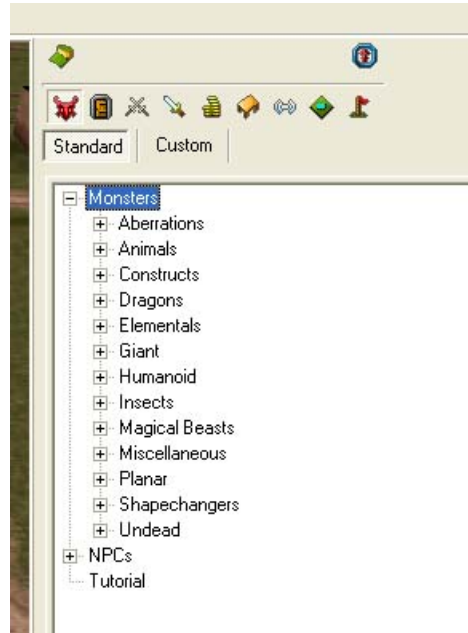
OBJECTS:

Image 1.1.3c – The Palette – Objects

Objects can be accessed from one of the 9 categories of objects (creatures, doors, encounters, items, merchants, placeables, sounds, triggers and waypoints) by enabling the appropriate button at the top of the Palette window.

Every element in the objects palette of the toolset is identified as a “Blueprint”. These blueprints represent templates of objects that, in general, can be used without any need for customization.

To place an object in your area, choose the appropriate blueprint in the palette (the “+” and “-” signs allow the lists of blueprints to be expanded or compressed) and then click in the Main View Window where you want it to be added.



All object categories are divided into “Standard” and “Custom” blueprints. The standard blueprints are those that were created by Bioware and that are used in the official campaigns. The custom ones are those that you will be creating specifically for your module.

To rotate an object you have placed in an area you use the buttons identified by curved red arrows at the bottom of the Main View Window.

The rest is basically a question of design. By using the tiles and objects that are available in the Palette, you can shape your world. You will find in the next chapters many designing tips to help you make the right choices in building your module, but do not forget that the only true limit in designing your module will be your imagination. For more information on designing and planning your module see [The Guide to Building Volume II the Design Manual](#).

CHAPTER 2 Modules

Why open the toolset if not to create a module? In this chapter, we will guide you in the very simple steps to follow to create a new module. We will also provide you with detailed comments on the properties of a module, which can be customized to some extent to set some general game play parameters.

2.1 Creating a New Module

Creating a new module is probably the simplest task you will have to perform in your module building experience.

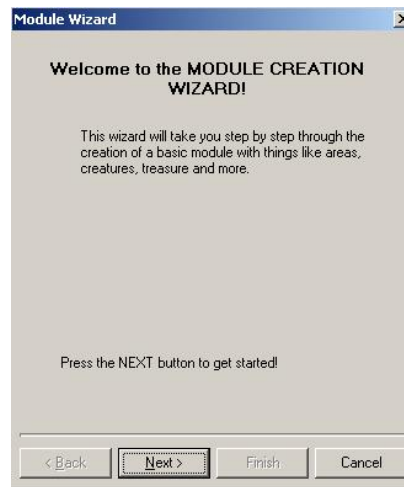
When you launch the toolset, you will immediately be asked if you want to create a new module. By selecting this option (instead of choosing to open an existing module), you will go through 8 basic steps to create your new module.

STEP 1:

You see a welcome screen. But do not be fooled by this warm and promising welcome! The module that you will create with the module creation wizard will NOT have creatures, treasures and more. These are all things that you will have the opportunity to add to your module at a later stage in the building process, but the module that will be created at the end of this creation wizard will in fact be plain empty. Consider yourself warned.

Now that the truth has been told, simply press the NEXT button and get to the next step.

Image 2.1a – Module Wizard – Step 1

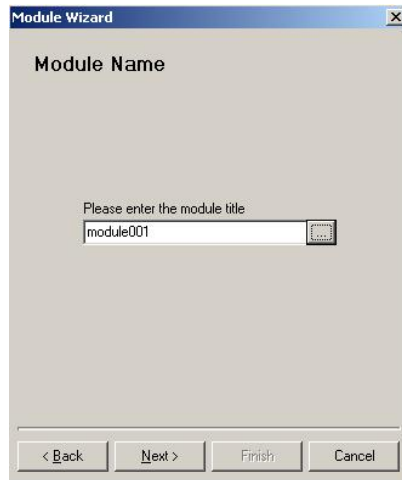


STEP 2:

Here you choose a name for your module. Do not be troubled if you do not find a cool name at this stage since you can change the name of your module at any time.

By clicking on the “...” button you will launch the “Edit String” window that permits you to add the translated names of your module in all the languages that are supported by NWN. Unless you want to build a module that will be translated in various languages, there is really no reason to press this “...” button.

Image 2.1b – Module Wizard – Step 2

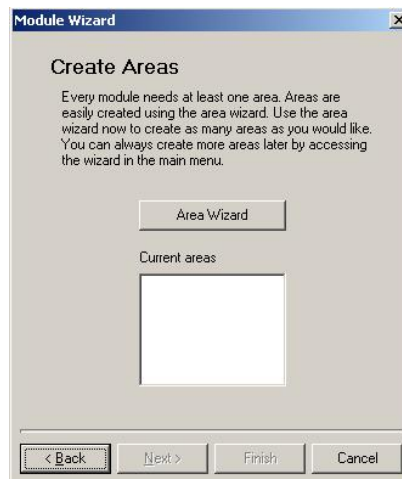


STEP 3

As mentioned previously, every module must have at least one area. By clicking on the “Area Wizard” button, you will launch the area creation tool that will permit you to create this first area.

This step is only a formality. You will have the opportunity to create many more areas in your module and to change the starting point of the players that will be playing your module. Do not therefore feel obliged to create as your first area the area where you intend the players to start.

Image 2.1c – Module Wizard – Step 3

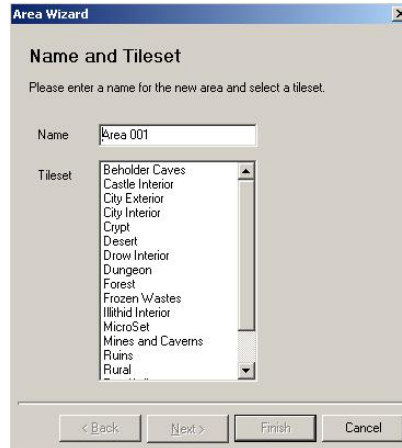


STEP 4:

This is where the first area of our mod will get its name. Again, naming things in the toolset is never permanent so do not bother to find the right name if you have not yet thought of one. You will have the opportunity to change it when your inspiration hits you.

The only important decision you will be making here is which tileset to use. There are a lot of choices and your selection will pretty much control the type of area you will be making. The tileset you choose in fact determines what tiles (features, groups and lands) that will be available when you design an area (see [Sub-Section 1.1.3](#)). Above all, note that the tileset you choose CANNOT be changed afterward. If you choose the wrong one you will have to create a whole new area to replace this one.

Image 2.1d – Module Wizard – Step 4

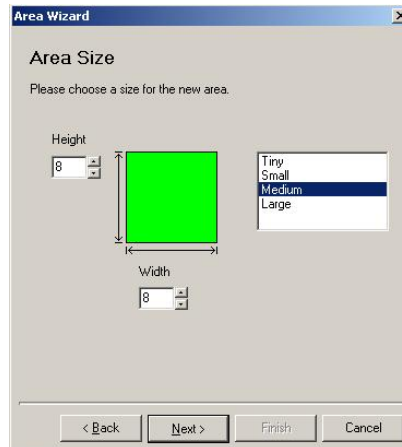


STEP 5:

Here you determine the size of your area. The unit in the "Height" and "Width" properties is the number of tiles.

This is one time where size does matter. While the game allows for areas as large as 32 by 32 (that's a total of 1024 squares/tiles), most computers have very long load times for areas that size. It is usually recommended to keep your areas to a maximum of 144 squares total (or something like 12 by 12). Keep in mind however that you can change the size of a selected area at any time, so you will have the opportunity to increase or reduce its size as you go along.

Image 2.1e – Module Wizard – Step 5



STEP 6:

You have now created your first area.

If you want to see and customize the properties of the area you have just created, you can check the "Launch Area Properties Dialog" box and click on "Finish". However, since these properties are customizable at any time, it is not necessary for you to edit them at this stage. If you do edit them or if you want to know what can be done with the properties of an area, then you might want to read [Section 3.2](#) of this manual.

Image 2.1f – Module Wizard – Step 6

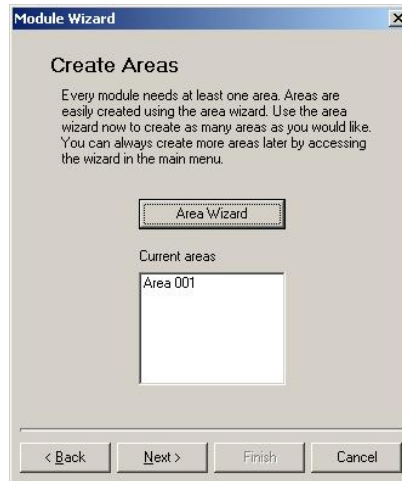


STEP 7:

This is it, the end of the Module Creation Wizard. You can now see your newly made area in the list of "Current areas" on the screen you first saw in Step 3 above. Clicking on the "Next" button will bring you to the final step of the creation of your new module.

If you have a very, very clear idea of what you want to create with this module and if you like things to get done at an early stage, then you can always click on the "Area Wizard" button again to add more areas to your module at this stage. However, since areas can be added at any time, it is unnecessary to create more empty areas now.

Image 2.1g – Module Wizard – Step 7



STEP 8:

Image 2.1h – Module Wizard – Step 8

Once you click on the "Finish" button appearing in this screen, you will be in the toolset proper.

If you are creating your very first module, you might want to read Chapter 1 of this manual to get acquainted with the general look of the toolset. The rest of this manual will provide you with detailed comments on all the functionality of the toolset to help you to make the best use of all the possibilities that are offered by this wonderful tool.



2.2 The Properties of a Module

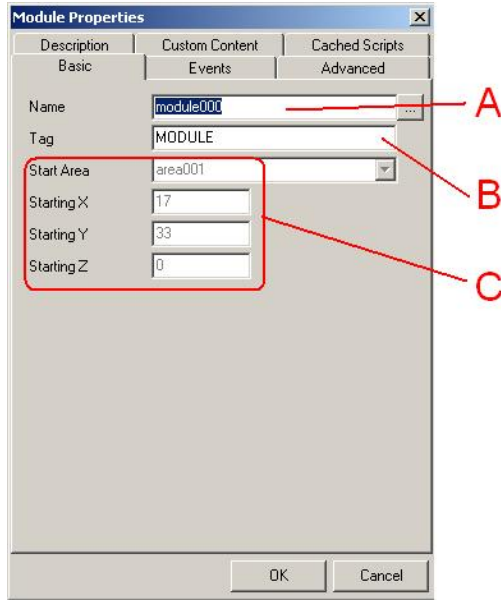
Your module has many properties that determine game rules (like resting and experience point allocation) and game play parameters (like the length of a day in your module).

To access your module's properties, go to the "Edit" menu of the toolset and select the "Module Properties" option.

2.2.1 The Basic Properties of a Module

When you open the module's properties screen, you will get the following window:

Image 2.2.1 – Module Properties Screen – The Basic Tab



- A** Name: Here you can change the name of the module. This is the name that players will see when they start a new NWN game, and are asked to select a module.

Note that the name of the module is different from the name of your module file (the file name you see when you open your modules NWN folder). There is no way to change the module file name in the toolset.

By clicking on the “...” button next to the name of your module, you will launch the “Edit String” window that permits you to add the name of your module translated into all the languages that are supported by NWN. Unless you want to build a module that will be translated in various languages, there is no reason to press this “...” button.

- B** Tag: Every object in NWN has a tag. The tag permits you to find/designate a specific object in a script. Since it is possible to designate your module as an object in a script with a `GetModule()` command, there is no reason at all to modify the module’s tag or to call it in a script by its tag. It is therefore recommended not to change the module’s tag in order to avoid duplication of tags.

- C** Start Area / X / Y / Z: These are the settings confirming where players will start when they load your module. It is not possible to change these parameters in the module’s properties screen. In order to change the starting location, you need to select the circle with a red arrow icon in the top right corner of the Palette and place this icon representing the starting location where you wish in your module, just like any other object.

2.2.2 The Events Properties of a Module

The events are the scripts that will run when specific events are triggered, generally by actions taken by players. In the following screen you will see a list of these possible events and the default scripts that are assigned to these events when a new module is created.

Image 2.2.2 – Module Properties Screen – The Events Tab



Modifying any of these events requires scripting knowledge. Although it is not within the scope of this manual to teach you how to script, here are a few comments that you must keep in mind when you modify these scripts. You might also want to read the [NWN Lexicon](#) on these events for more details and examples of scripts. (Many of the comments below are taken from the Lexicon).

OnAcquireItem: The script attached to this event fires when an item is acquired; whether by a player or by a creature. **This script fires on creature spawn for each item held in the creature's inventory.** This script is therefore very taxing on the engine as it can be fired very, very often.

This script is usually designed to get the tag of the item that is acquired and then test it against a list of tags to determine if a special event must be fired.

The default script assigned on module creation provides that, when you set an integer variable named X2_SWITCH_ENABLE_TAGBASED_SCRIPTS on the module and you give it a value of 1 (see [Sub-Section 2.2.3](#) about module variables and [Chapter 16](#) to learn more), then each time an item is created the engine will run the script that has the same name as the tag of the item (if such script exists). Needless to say that this can be very taxing on the game engine.

OnActivateItem: The script attached to this event fires when an item having either the item property spell "Unique Power" or "Unique Power - Self Only" is activated by a player. This is therefore where

you script the special powers you intend to give to custom items you have created for your module. See Section 7.3 for more details on how to add these item properties to an item. This event is also called when the scripting command `EventActivateItem()` is used in a script.

This script is usually designed to get the tag of the item that is activated and then test it against a list of tags to determine what special effect must be fired.

The default script assigned on module creation provides that, if you set the variable `X2_SWITCH_ENABLE_TAGBASED_SCRIPTS` on the module to 1 (see Sub-Section 2.2.3 about module variables and Chapter 16 `x2_INC_SWITCHES` to learn more), then each time an item is activated the engine will run the script that has the same name as the tag of the item (if such script exists).

OnClientEnter: The script attached to this event fires when a client (player or DM) enters the module. This script will generally be used in multiplayer or permanent world modules to perform some initialization of variables for a particular player or to reset encounters.

When a PC enters a module they the charater is fully loaded yet. The PC also 'acquire' all of their items and fire the module `OnAcquire` script for each item. `AssignCommands` don't work on them because they aren't in an area yet, but they are a proper object. This can cause unpredictable behaviour. The best place to try to manipulate the PC using an `OnEnter` event is when they are entering an area and not the module. This is more relieable and should make trouble shooting your scripts easier.

OnClientLeave: The script attached to this event fires whenever a player logs out of a server or leaves the module. This script will generally be used in multiplayer or permanent world modules to store information on the module or to re-initialize variables.

OnCutsceneAbort: This event fires when a player tries to cancel a cutscene by pressing the "ESC" button. This script is usually designed to set a variable either on the player or on the module when the cutscene starts to establish which cutscene is actually run. Then, when the `OnCutsceneAbort` event fires, the script checks for this variable and, through a switch case statement, establishes the series of commands that must be performed when the player wants to skip a cutscene.

Note that if a player hits the "ESC" button while in a cutscene, nothing will happen except for this event to fire. You'll have to script everything yourself to take the player out of the cutscene, including clearing the action queues of all cutscene objects, setting `SetCutsceneMode()` to `FALSE`, and removing invisibility effects from the player if applied.

OnHeartbeat: The script associated with this event fires every six seconds. This can be very taxing on your engine.

OnModuleLoad: The script attached to this event fires every time the module is loaded, including when loading a saved game. This is a good place to initialize any variables that need to be initialized. Keep in mind however that action on a player can only be performed once they are in an area included in your module. Therefore, this script should not be used to assign any commands to players.

The default script assigned to this event is basically a template script that provides you with comments and suggested script functions to turn on the switches that are more fully described in this manual in Sub-Section 2.2.3 regarding module variables and Chapter 16 `x2_INC_SWITCHES`.

OnPlayerDeath: The script attached to this event fires when a player has been reduced to -10 hit points or less. Unless you change the default script assigned to the `OnPlayerDying` event, your module will not perform any "dying" scripts such as you see in some custom made modules, and will consider a player to be dead once their hit points get to 0 or lower.

The default script assigned to this event will clear the relationships of the standard factions to the player and determines when the "Death GUI Panel" that allows the player to exit NWN or respawn will appear.

OnPlayerDying: The script attached to this event fires when the player has been reduced to 0 hit points, but not below -10 (this state is considered unconscious).

The default script attached to this event will apply an `EffectDeath()` effect to the player to kill him quickly and fire the `OnPlayerDeath` event. If this script is removed, the player remains disabled indefinitely unless of course you replace this script with some form of bleeding system script.

OnPlayerEquipItem: The script attached to this event fires whenever a player equips an item. It can be useful in restricting the use of an item in certain areas, for example.

This script is usually designed to get the tag of the item that is equipped and then test it against a list of tags to determine what conditions or events must be fired.

The default script attached to this event seems to check for special items that are used in the HotU campaign (namely an intelligent weapon and another one that drains the player). Furthermore, this default script provides that, if you set the variable `X2_SWITCH_ENABLE_TAGBASED_SCRIPTS` on the module to 1 (see Sub-section 2.2.3 about module variables and Chapter 16 `x2_INC_SWITCHES` to learn more), then each time an item is equipped the engine will run the script that has the same name as the tag of the item (if such script exists).

OnPlayerLevelUp: This script fires when a player has gained a new level. It fires right after the player has completed the levelling up process. There is no way to prevent the level up process from occurring (except of course by preventing players from gaining any experience in the module), but you can script it so that the player is knocked back a level right after he has gained it.

OnPlayerRespawn: The script attached to this event fires when the player selects the "Respawn" button after dying. This event is therefore often used to teleport the player back to a temple and hit them for an XP and/or gold penalty.

If there is no script attached to this event, the player will respawn where he or she died with no penalties, and be returned to full health.

The default script assigned to this event resurrects the player character, removes all adverse effects and applies an XP and gold penalty. This script checks if there is a waypoint tagged `"NW_DEATH_TEMPLE"` and, if so, teleports the player to that waypoint. If there is no `"NW_DEATH_TEMPLE"` waypoint, the PC is resurrected where he fell.

OnPlayerRest: The script attached to this event fires: (1) when a player attempts to rest; (2) when a player cancels resting; AND (3) when resting is finished. The script will therefore fire twice in a normal resting process (at the beginning and at the very end).

It can be used to spawn some creatures around the player to prevent a successful rest in some areas, or to prevent natural healing altogether by bestowing a curse upon the player.

You do not need to script this event if you do not want players to be able to rest in specific areas, since each area in its advanced properties can have the resting option disabled (see Sub-Section 3.2.5).

The default script assigned to this event checks if the variable X2_SWITCH_ENABLE_XP2_RESTSYSTEM is set to 1 and, if the appropriate variables are set for each area runs the HotU Wandering Monster System during the resting process (see Sub-Section 2.2.3 about module variables and Chapter 16 x2_INC_SWITCHES to learn more).

OnPlayerUnequipItem: This script fires whenever a player unequips an item. This can be useful for creating cursed items that cannot ever be unequipped.

This script is usually designed to get the tag of the item that is unequipped and test it against a list of tags to determine what conditions or events must be fired.

The default script attached to this event provides that, if you set the variable X2_SWITCH_ENABLE_TAGBASED_SCRIPTS on the module to 1 (see Sub-Section 2.2.3 about module variables and Chapter 16 x2_INC_SWITCHES to learn more), then each time an item is unequipped the engine will run the script that has the same name as the tag of the item (if such script exists).

OnUnAcquireItem: The script attached to this event fires when an item is removed from a creature's inventory (player characters or NPCs), whether because the player drops it or because he sells it. It can be useful to place the item in a common location (such as the Divining Pool in the original campaign) when the item is related to the plot.

This script is usually designed to get the tag of the item that is dropped and test it against a list of tags to determine what conditions or events must be fired.

The default script attached to this event provides that, if you set the variable X2_SWITCH_ENABLE_TAGBASED_SCRIPTS on the module to 1 (see Sub-Section 2.2.3 about module variables and Chapter 16 x2_INC_SWITCHES to learn more), then each time a player loses an item the engine runs the script that has the same name as the tag of the item (if such script exists).

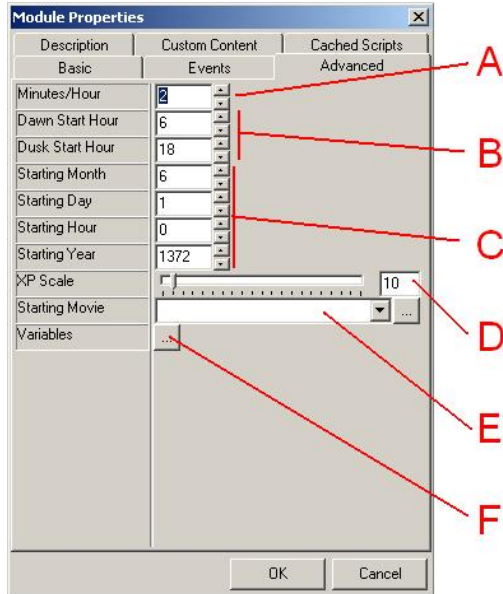
OnUserDefined: The script attached to this event fires when events are signalled through a script on the module.

Typically this script is designed as a switch/case statement that does something different depending on the signal (numerical value) sent to it by an event that calls the script.

2.2.3 The Advanced Properties of a Module

You will notice while editing the properties of the various objects that will be populating your module that each category of objects has its "Advanced" properties. What is supposed to be advanced is in fact just a melting pot of properties that do not belong in any other property categories. This is no different for the advanced properties of a module as you will see in the following screen:

Image 2.2.3a – Module Properties Screen – The Advanced Tab



- A** Minutes/Hour: Here you can establish how many minutes of gameplay it takes for the game to consider that one hour has passed.
- B** Dawn/Dusk Start Hour: It is here that you can set at what hour the day and night cycle will change in the game.
- C** Starting Month/Day/Hour/Year: By setting these, you can set the date and hour at which the module starts when it is loaded. Of course, the day/month/year setting will usually only be important story-wise, but the hour setting may have a great importance if you wish the players to start during the night or during daylight.
- D** XP Scale: This scale determines, in percentage, how many experience points you want the players to obtain when killing a creature, versus the amount of points they would receive according to the Dungeon & Dragon core rules. For example the NWN Official Campaign is set to 10% meaning that each creature killed rewards the player 10% of the possible total experience for a creature of that Challenge Rating.
- E** Starting Movie: By clicking on the "..." button you will get the list of movies that are available in the movie folder of your NWN game.

There is a very detailed tutorial on how to make your own starting movies on the BioWare website at the following address:

http://nwn.bioware.com/builders/movie_tutorial.html

Note that if you want players to be able to see your movie, it will be necessary for them to have the movie file in their movie folder as well.

- F** Variables: You can set local variables on the module by clicking on the “Variable” button. These variables can be used in any script where the module is used as an object. For more information on these and other predefined variables see and Chapter 16 x2_INC_SWITCHES.

Some specific variables have already been coded by BioWare and can be used to modify the properties of your module. Here is a list of these already coded variables:

- i) X2_SWITCH_CROSSAREA_WALKWAYPOINTS
If set to 1, NPCs walking between waypoints using the WalkWayPoints() command will cross areas when their walking route brings them in multiple areas. Otherwise they will stay in the area they are spawned in.
- ii) X2_L_NOTREASURE
If set to 1, you will disable on your module the random loot generation system used in the creatures' OnSpawn script.
- iii) X2_SWITCH_ENABLE_UMD_SCROLLS
If switched to 1, a rogue has to succeed in a Use Magic Device check against DC 25+SpellLevel in order to use a scroll.
- iv) X2_SWITCH_DISABLE_ITEMCREATION_FEATS
If set to 1, the Item Creation Feats that come with Hordes of the Underdark will be disabled.
- v) X0_G_ALLOWSPELLSTOHURT
If set to 1, area of effect spells will hurt NPCs that are neutral to the caster if they are caught in the effect.
- vi) X2_SWITCH_ENABLE_50_WAND_CHARGES
If set to 1, the crafting system will create 50 charges on a new wand instead of 10 + casterlevel charges.
- vii) X2_SWITCH_EPIC_SPELLS_HURT_CASTER
If set to 1, some epic spells, such as Hellball, will do damage to the caster.
- viii) X2_SWITCH_SPELL_CORERULE_DMTOUCH

- If set to 1, Deathless Master Touch will no longer affect creatures of large size and above.
- ix) X2_SWITCH_RESTRICT_USE_POISON_FEAT
- If set to 1, only the player characters with the Use Poison feat will be able to use the various poisons that can be found to poison their weapons. Otherwise, all characters can use any type of poisons as long as they are successful in a dexterity check.
- x) X2_SWITCH_MULTI_HENCH_AOE_MADNESS
- If set to 1, henchmen will be able to damage each other with area of effect spells and potentially go at each other's throats. Warning: Activating this switch has the potential of introducing game breaking bugs.
- xi) X2_SWITCH_ENABLE_NPC_AOE_HURT_ALLIES
- If set to 1, creatures will hurt each other with their area of effect spells even if there are no player characters in the area of effect.
- xii) X2_SWITCH_BEBILITH_HARDCORE_RUIN_ARMOR
- If set to 1, the Bebilith Ruin Armour ability is going to actually destroy the armour it hits.
- xiii) X2_SWITCH_GLYPH_OF_WARDING_INVISIBLE
- If set to 1, the Glyph of warding symbol will disappear after 6 seconds, but the glyph will stay active.
- xiv) X2_SWITCH_DISABLE_SECRET_DOOR_FLASH
- If set to 1, the glow of a newly found secret door used in some locations in HotU will be disabled.
- xv) X2_SWITCH_ENABLE_TAGBASED_SCRIPTS
- If set to 1, the tagbased scripts that are available when using the standard module events (x2_mod_def_*) will be enabled.
- xvi) X2_SWITCH_ENABLE_XP2_RESYSTEM
- If set to 1, and if you are using the default rest script and if the appropriate variables are set for each area, then it will enable the HotU Wandering Monster System for the module.
- xvii) X2_L_AI_NO_AOE_DISPEL
- If set to 1, then the artificial intelligence will not use Dispel Magic against harmful area of effect spells.
- xviii) X2_L_STOP_EXPERTISE_ABUSE

If set to 1, the Expertise/Improved Expertise modes will be disabled whenever a player is casting a spell.

2.2.4 The Description of a Module

The Description tab permits you to type the description you want a player to see when they are starting a new module. It is a good idea to add in your description not only a synopsis of the story of the module, but also the module requirements like the class, level and alignment restrictions and any other information players must know before starting to play the module.

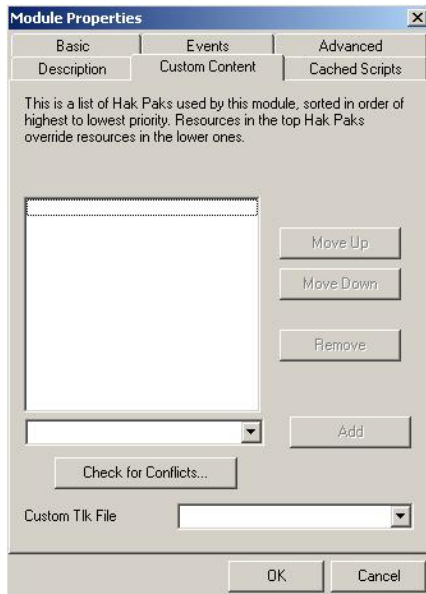
The “...” in the lower right corner of this screen permits you to launch the “Edit String” window in which you can add the translated description of your module in all the languages that are supported by NWN. Unless you want to build a module that will be translated into various languages, there is really no reason to press this “...” button.

2.2.5 The Custom Content of a Module

The toolset provides you with a great variety of tilesets, creatures and placeable objects (just to name a few categories) to choose from. However, there might come a time when the standard tilesets or objects do not provide the building or creature you are looking for to design the world you want to create in your module. This is when hakpacks come into play.

A hakpack is a file containing additional resources which can be added to the module to which it is attached. You can add hakpacks to your module in the “Custom Content” property screen.

Image 2.2.5a – Module Properties Screen – The Custom Content Tab



The most popular hakpack is clearly the Community Expansion Pack (“CEP”) that in fact is the amalgamation of a great variety of custom made content by various authors in the NWN community. The CEP hakpack and full details on how to install it may be found here:

<http://nwwvault.ign.com/cep/>

Besides CEP, you will also be able to find a multitude of community created hakpacks on the NWVault website at the following address:

<http://nwwvault.ign.com/Files/hakpacks/>

To add a hakpack to your module, you must first download it and place it in the “hak” folder of your NWN game. You will then be able to see this hakpack in the file list appearing next to the “Add” button in the Custom Content screen. Then, by clicking on the “Add” button, the hakpack will be added to the resources used by the toolset for your module. You will notice when you click on the “OK” button that the module will re-build itself, which can be a very long process depending on the size of your module. This is necessary to adjust your module and the Palette so that it can use the newly available resources.

Most non-tileset hakpacks will come with an “erf” file that can be imported into your module by choosing the “Import” option found in the “File” menu of the toolset. These “erf” files contain the blueprints that will be added to your Palette and allow you to add the new resources included in the hakpack without any need for you to create your own blueprints. If the hakpack is only adding new tilesets to your module, then such “erf” files are not necessary and the new tiles will be added automatically to your palette. If however the hakpack is not a tileset hakpack and there are no “erf” files provided by the author, then you will have to create your own blueprints by using, (if as an example the hakpack provides new creatures), the new creature appearances coming with the hakpack that can be found in the file list of all possible appearances.

Adding multiple hakpacks can become very tricky. When hakpacks use resources with the same name, then the hakpack that is on top of your hakpack list will override the conflicting resources of the other hakpacks. There are no easy ways to solve this problem. You will usually have to edit some 2da files at that point, which requires fairly advanced knowledge about game resources. Such operations are not in the scope of this manual.

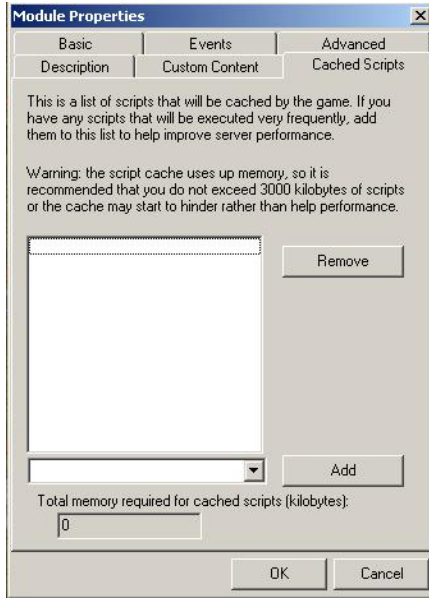
It is important to keep in mind that, from the moment you add a hakpack to your module, every player will need to have this hakpack in their hak folder to be able to play your module. You must therefore provide players with sufficient information for them to be able to find and download the hakpacks you have used.

2.2.6 The Cached Scripts of a Module

Cached scripts are kept in the engine’s memory. They may therefore be accessed more rapidly. It is a good idea to cache all scripts that are used very frequently in your module to improve the module’s performance and, hopefully, to avoid any in-game lag.

To add cached scripts for your module you must access the tab provided for that purpose in the Module Properties screen, as shown in the following image:

Image 2.2.6a – Module Properties Screen – The Cached Scripts Tab



So, you are probably asking yourself which scripts you should add to the cache. Here is part of the answer, provided by Bioware's Georg Zoeller:

How to decide which scripts to cache?

When you try to decide which scripts to use with the new script caching feature, you should know how it works:

The function allows you to cache compiled scripts (.ncs) into RAM, thus speeding up their execution time;

Since it caches compiled scripts, it makes no sense to cache include files, as they are not compiled

How many scripts you should/can cache is dependent on your computers RAM. Please keep in mind that this is stored on the module, so if you create modules for the public, you should remember that many people's computers may have less RAM than you have at your computer

It makes sense to cache the most used scripts in your campaign, so heartbeat scripts, spawn in scripts for often-used monsters, etc make sense to cache, while caching an OnRest event script may not yield any noticeable performance increase in your module.

And more from Georg:

In general, I would cache:

Creature event scripts, especially nw_c2_default1 and nw_c2_default9, the module's OnAcquire and OnHeartbeat events and maybe OnClientEnter and OnItemActivated as well. With Hordes you should definitely add x2_s3_onhitspell.ncs to that list if you add any code to it.

It makes no sense to cache things that are only run very infrequently (i.e. OnUse scripts for placeables, most conversation scripts, etc)

2.3 Designing Tips

Do not build too many areas in your module: There are no limits to the number of areas that can be included in your module. Keep in mind however that the more areas there are, the longer it will take for players to load the module and to load a saved game after they die. Furthermore, NWN was not designed to create huge worlds and any module with more than 100 areas is likely not only to have performance issues, but also to create uncontrolled freezes for players. If your intention is to build a very large module, then it might be a good idea to split this module in many shorter ones and create a campaign.

If you use hakpacks, stay reasonable as to their size: Hakpacks are a wonderful resource to the game. The community has managed to achieve very high quality standard in building many, many custom made hakpacks. Keep in mind however that adding many or large hakpacks to your module might drive players away. If it is necessary to download 100 MB worth of hakpacks to play a 2-hour module, you can expect your module to stay on the shelves.

Close your module in the toolset when you test it: To test anything you have built in your module, you will need to play the game (and probably to adjust the starting location so that you test only the part that needs to be tested). When you do that, it might be important to close your module in the toolset if you have other software applications opened. Playing your module while it is still opened in the toolset will sometimes corrupt the module, amongst many other possible causes, insufficient memory or virtual memory allocation. Bottom line, if you are not sure that your computer can handle all the applications that remain opened while testing your module, then close it in the toolset to avoid any corruption.

Save often and make back-up copies of your module: It is a known problem that custom made modules will sometime get corrupted, usually for no apparent reason. Making back-ups will save you a lot of time and frustration if this ever happens to you.

2.4 How to...

2.4.1 ...have multiple starting locations in your module.

Only one starting location can be placed in your module. However, you can always redirect players from this starting location to other places depending on their class, alignment, race or any other criteria by either placing a trigger where the starting location is or by creating a starting room where a NPC will redirect players to the appropriate location. Whatever the method you choose you will need to write a script that will first check for the criteria that you want to be used to determine where each player should start and then to make the players jump to their respective starting locations.

2.4.2 ...have the player characters respawning after death in a specific location.

The easiest way to do this is simply to place a waypoint where you want the players to respawn and to assign the tag NW_DEATH_TEMPLE to this waypoint. This will however only work if you have left the "nw_o0_respawn" default script in the OnPlayerRespawn event of your module. (See Sub-section 2.2.2 for more comments about this event.)

If you want multiple respawning locations, this can only be done by modifying the OnPlayerRespawn event of your module.

CHAPTER 3 AREAS

3.1 *Creating an Area with the Area Wizard*

The Area Wizard is launched automatically when you create a new module, to create a first area. You are therefore probably already familiar with the process of creating new areas.

If you want to add more areas to your module, you can do it by:

- a) selecting the option “Area Wizard” in the “Wizards” menu of the toolset; or
- b) right-clicking on the title “Areas” in the module structure tree and choosing the option “New”.

The process thereafter followed to create the new area was already covered above in the module creation process. Refer to steps 4 to 6 of Section 2.1 above for more details on this process.

3.2 *The Properties of an Area*

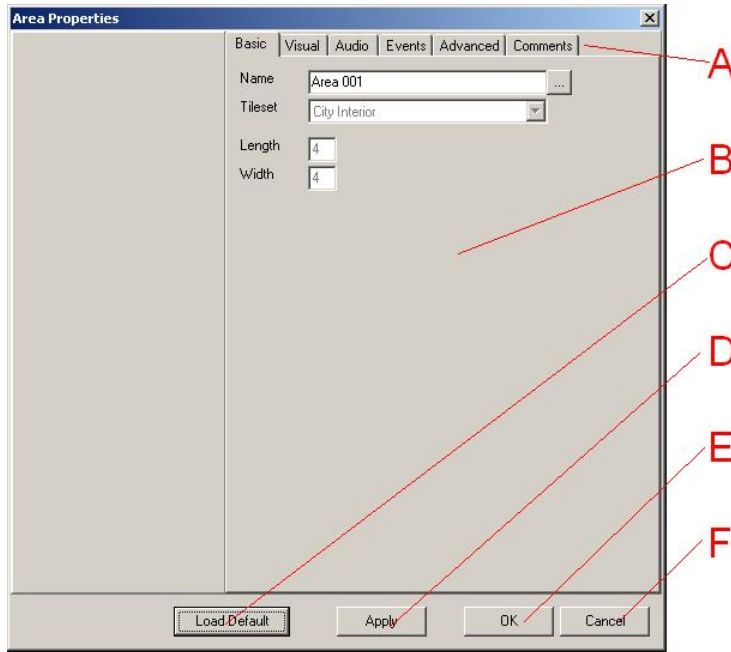
It is in the area properties that you can customize an area. Area customizing plays an important role in creating atmosphere for your module and this should therefore be done with great care.

In order to do so, you will have to access the properties of an area by either:

- 1) Right-clicking on the area and select the “Properties” option in the menu that appears; or
- 2) Selecting the area and, in the “Edit” menu, selecting “Area Properties”.

The following screen will then appear on your screen:

Image 3.2a – Area Properties Screen



- A** These are the category tabs of the area properties. Selecting any one of these category tabs will show the current area properties on the related subject in the information window designated by the letter “B”. These tabs are:

The “Basic” tab (see Sub-section 3.2.1);

The “Visual” tab (see Sub-section 3.2.2);

The “Audio” tab (see Sub-section 3.2.3);

The “Events” tab (see Sub-section 3.2.4);

The “Advanced” tab (see Sub-section 3.2.5);

The “Comments” tab (see Sub-section 3.2.6);

- B** This is the information window in which you will find the area properties related to the tab that has been selected.
- C** Clicking on this button will reset all the area properties, in each one of the category tabs, to the default properties that were automatically set by the toolset when the area was created. If you have clicked on this button by mistake, then the only way to retrieve your area properties is to click on the “Cancel” button.
- D** Clicking on this button will apply the current modifications to the area and the results can then be seen or heard in the toolset window behind the area property screen. This option is particularly useful when you are modifying the “Audio” properties. Even after you have

applied the modification on the area, you still have the option of keeping or discarding these modifications (by using either the “OK” or “Cancel” button).

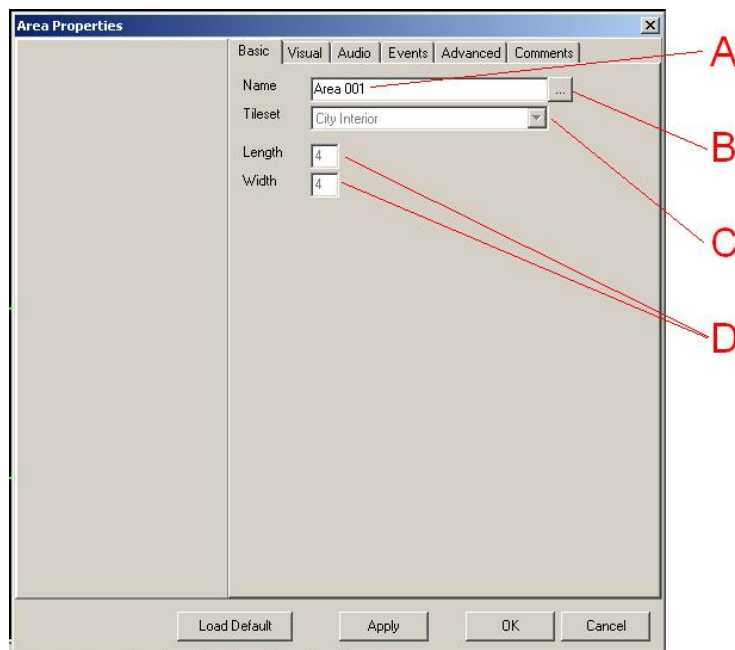
Note that the “Apply” button is only available as long as the area you are modifying is currently opened in the toolset.

- E** Clicking on this button confirms that you are satisfied with the modifications you have made to the properties of the area so that these modifications will be kept.
- F** Clicking on this button will cancel all the modifications you have made to the properties of the selected area since you have opened the Area Properties Screen.

3.2.1 The Basic Area Properties

The following screen appears when you click on the “Basic” tab of the area properties window:

Image 3.2.1a – Area Properties Screen – Basic Tab



- A** Name: This is the name of the area. Players will see this name on the area loading screen when they enter this specific area.
- B** “...” Button: Clicking on this button will open the Multilanguage window in which you can determine the name which will be appearing to players that are playing a foreign version of NWN.

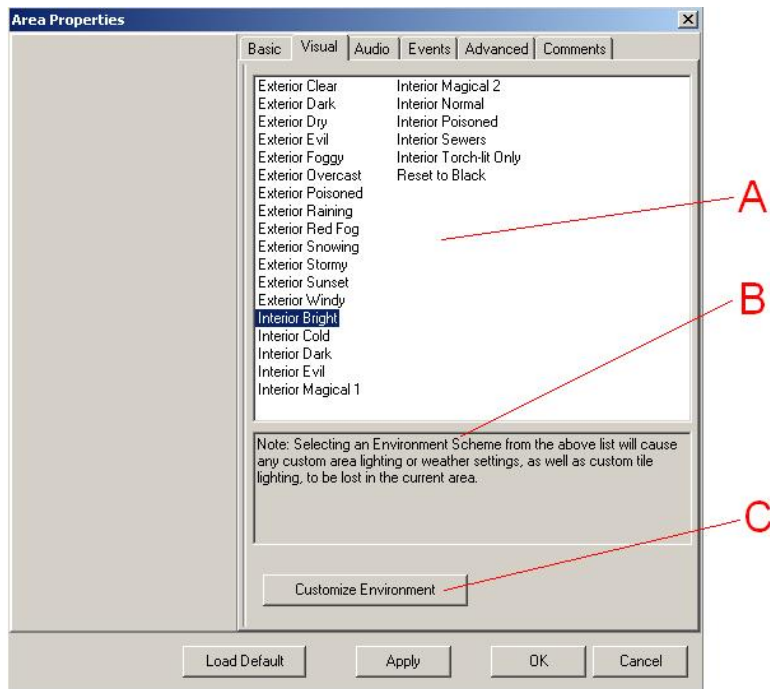
- C Tileset: This identifies the tileset type used for this area. It is not possible to modify this property. If you wish to change the tileset type, you have no other choice than to create a whole new area and delete this one.
- D Length/Width: These are the length and width (in number of tiles) properties of this area. You cannot change these properties through this window. In order to change these properties, you must exit the area properties window and select, in the “Edit” menu of the toolset, the “Resize Area” option.

3.2.2 The Visual Area Properties

1) Setting and Changing the Visual Properties of an Area

The following screen appears when you click on the “Visual” tab of the area properties window. These properties are those that were set by default when the area was created or, if you have changed them previously, those that you have selected.

Image 3.2.2a – Area Properties Screen – Visual Tab

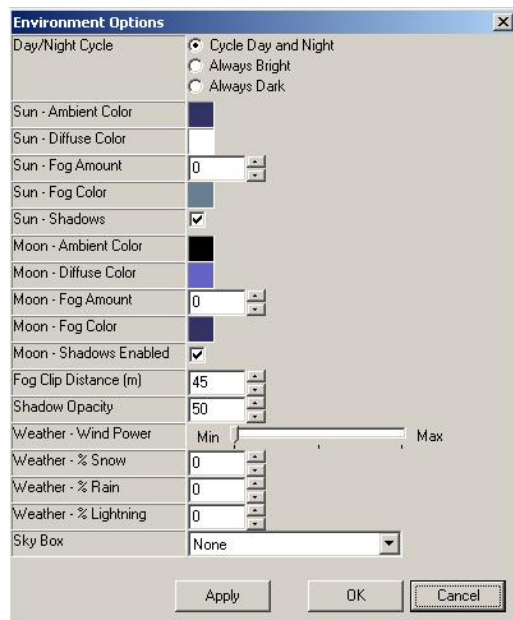


- A You can choose in this window any of the pre-determined Environment Schemes to fit with the type of area you wish to build. The names of these Environment Schemes are pretty accurate and you can therefore rely on the names when it is time to choose the appropriate environment for your area.

These schemes establish the color of the lighting, the quantity of fog, the color of the fog and the weather properties of the area, but not the tile lighting (see sub-section 3.4.2 for comments on how to change tile lighting). Once a general scheme has been chosen, it is always possible to customize any of the settings of this scheme by clicking the “Customize Environment” button (see “C” below).

- B** This warning is self-explanatory. If you click by mistake on another Environment Scheme, then the only way to avoid losing all your custom lighting or weather settings is by clicking on the “Cancel” button.
- C** Clicking on this button will open the “Environment” window, which can be used to customize the settings of the Environment Scheme you have chosen for this area. The Environment window of each one of the Environment Schemes will of course have different settings. Here is the window that will appear if you have chosen the “Exterior Clear” Environment Scheme:

Image 3.2.2b – Environment Options Screen



These settings have the following parameters:

Day/Night Cycle: This setting permits you to determine if you want the day and night settings to cycle in this area, or if you always want to keep the day or night settings. Changing the “Cycle Day and Night” setting to “Always Bright” will disable all “Moon” settings and choosing the “Always Dark” option will disable all “Sun” settings. If the “Cycle Day and Night” option is selected, the visual settings will be the ones applicable in the “Sun” properties during day and in the “Moon” properties during the night.

Ambient Color: The ambient color is reflected on the player characters, the objects, and some part of the terrain features. In other words, ambient lighting is like a spotlight that points straight

down on the area. This color setting therefore appears on placed objects, not in the air (i.e. it does not affect the color of the lighting of the sun or moon) or on the fog.

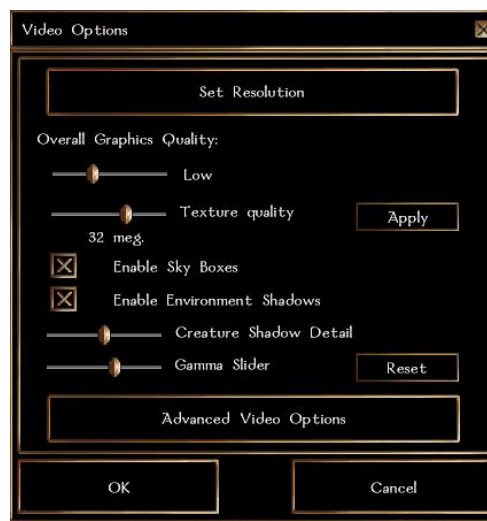
Diffuse Color: Diffuse is a glow that comes from the sides, sort of like the soft light you see in those fuzzy photographs. If you select a white or bright colored ambient light your area will be brightly lit from above. With a bright ambient light, everything is well illuminated. Use more subdued tones to get a darker effect. Note that the diffuse color does not reflect on the plain air, the fog or the objects in the area.

Fog Amount: The setting of this property goes from 0-200. It is unclear what unit is used to determine how much fog will be added in the area. One thing is certain, the greater the value is the more fog there will be in the area. A setting of 20 will often be sufficient to create a really foggy area. With a setting of 200, you can barely see the area.

Fog Color: This setting determines the color of the fog in this specific area and also the color of the air. Therefore, even if the Fog Amount is set to "0", changing the Fog Color setting will change the color of the sky and surrounding ambient air.

Shadows/Shadow Enabled: By ticking the shadows box, shadows will be created by player characters and by diverse objects and parts of tiles forming the environment of the area. However, if players have not checked the "Enable Environment Shadows" box as shown in the Video Options menu, only the shadow created by the player's character will be shown on the screen of this specific player.

Image 3.2.2c – Video Options Menu



Fog Clip Distance: The setting of this property must be set between 30 and 9999 meters. This setting determines how far from the player characters the fog seems to be. Higher settings may cause some players to experience "lag" or poor game performance.

Shadow Opacity: This setting must be between 1 and 100. The higher this setting is, the darker the shadows in the area will be. For this setting to have any effect, the Shadows box must be ticked (see above).

Wind Power: This setting determines whether your hair moves in the wind and how much grass sways (unless turned off by the player). Also, the wind ambient sounds will be more present and audible when this setting is set to the maximum, whatever the video options of the player are set

to. This setting cannot be changed if the area is designated as an indoor area (see the “Advanced” area properties in Sub-section 3.2.5);

% Snow / % Rain / % Lightning: These settings determine the chances that this area will be affected by one (or many since lightning may be used in conjunction with either rain or snow) of these weather features. This check is made when the first player character enters this area. It is unclear if this check is made again at some point if the player does not leave the area for a long time. If both % snow and % rain are set to 100, rain seems to have precedence over snow and it will always rain in the area. These weather conditions also automatically come with some ambient sounds when they are present in the area. These settings cannot be changed if the area is designated as an indoor area (see the “Advanced” area properties in Sub-section 3.2.5).

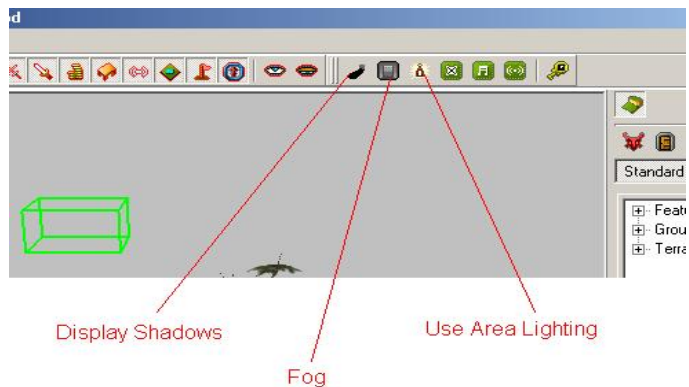
Sky Box: Here you can choose how the sky will appear to players. This neat little feature that was added with HotU has to be controlled manually every time you want to use it. Note that players must have the sky box option enabled in their Video Options panel (see menu appearing as Image 3.2.2c) to be able to see the chosen sky.

2) Testing the Visual Properties of an Area

After you have changed the parameters of the visual properties of the area, there are only two options for you to see the results of your modifications:

- 1) By clicking the “Display Shadows”, “Fog” and “Use Area Lighting” buttons on the toolbar of the general window of the toolset as shown in the image 3.2.2d, you will have a very, very preliminary look at the visual properties of your area. You should however not rely on this first look unless you simply wanted to test some colors since the visual area properties as seen in the toolset will differ a lot from what the player will actually see.

Image 3.2.2d – Toolset General Window



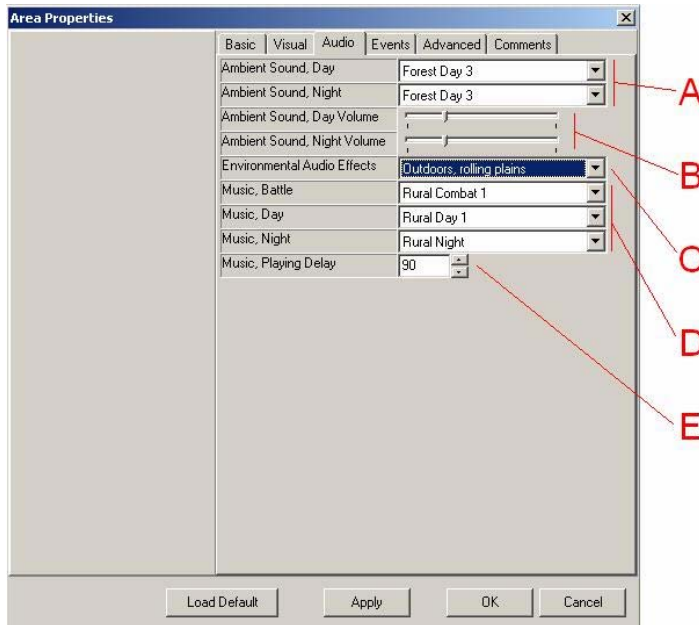
- 2) By play-testing the area. There is no better way to see the visual properties of an area than by playing the module.

3.2.3 The Audio Area Properties

I – Setting and Changing the Audio Properties of an Area

The following screen appears when you click on the “Audio” tab of the area properties window. These properties are those that were set by default when the area was created or, if you have changed them previously, those that you have selected.

Image 3.2.3a – The Area Properties Screen – Audio Tab



- A** Ambient Sound, Day/Night: This setting allows you to choose between a panoply of pre-determined categories of ambient sounds that can be assigned to the area. You can select different sets of ambient sounds that will be played during the day and during the night in this specific area.

Be aware that sounds produced by any weather condition (see “%Snow / %Rain / %Lightning” video property of the area in Sub-section 3.2.2) will override these ambient sounds when the weather condition affects the area.

It is not recommended that you rely entirely on these settings to create the audio atmosphere of your area. Placeable sounds (see Chapter 11) should be used to polish the audio properties of your area.

- B** Ambient Sound, Day/Night Volume: By this sliding scale you can set the volume of the ambient sound category you have chosen for this specific area. Setting it to the maximum will make the ambient sounds approximately as audible as the music that is played in the area.

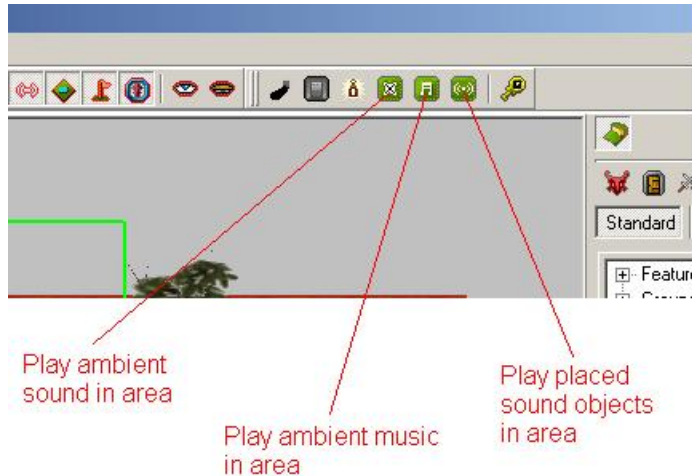
- C Environmental Audio Effects: The effects of this setting are unclear. The sounds of the environment properties of the area do not seem to be affected in any way by this audio setting, although the performance of these audio effects may be related to the sound card and audio settings of the player.
- D Music Battle/Day/Night: Here you can determine what music will be playing depending on the hour of the day. Choose “No Music Track” in these settings if you do not want any music to play. The battle music will start when a player character is attacked and will override the day or night music in the area.
- E Music Playing Delay: This setting determines how many seconds the engine will wait after the music has finished playing before starting it over again.

II – Testing the Audio Properties of an Area

The audio properties of your area can easily be tested by:

- a) clicking the “Play ambient sound in area”, “Play ambient music in area” and “Play placed sound objects in area” button on the toolbar of the general window of the toolset as shown in Image 3.2.3b. You will then have a very good idea of the sound effects and music playing in the area. The only sound effects you will not hear will be those created by the weather conditions in the area (see “%Snow / %Rain / %Lightning” video property of the area in Sub-section 3.2.2).

Image 3.2.3b – Toolset General Window



- b) play testing the area. There is no better way to hear what the players will hear when they will be playing your module.

TIP: It is a good idea to leave the three buttons shown in Image 3.2.3b enabled when you modify the audio options of the area you are working on. That way, you can hear the results of your modification even when the “Audio Properties” screen is still opened by clicking on the “Apply”

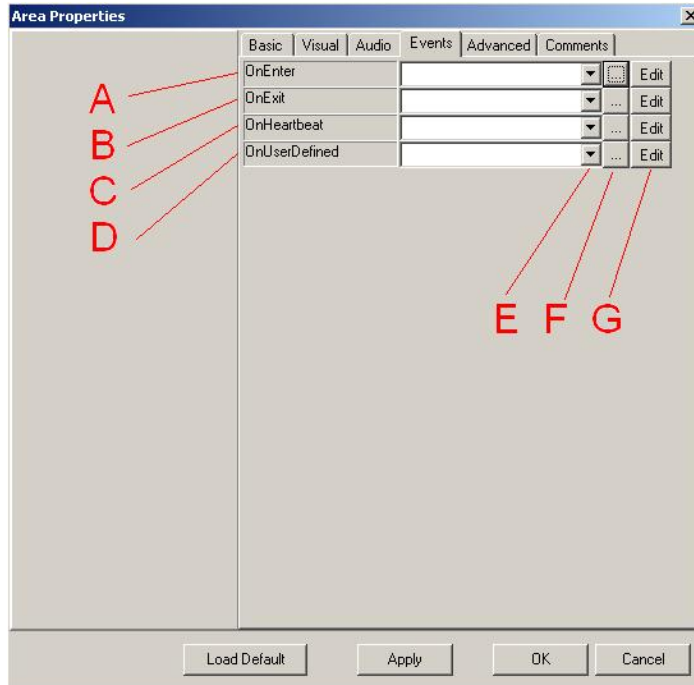
button of this screen. If you are unhappy with the modifications you have made, then simply click on the “Cancel” button on the screen to remove them.

TIP: The music played in the area when the “Play ambient music in area” button is enabled is the day music. Therefore, when you want to choose the night or combat music for a the area opened in the toolset, test it first as the day music to hear it and then assign it to the night or combat music setting when you have found the day music that you were looking for.

3.2.4 The Events Area Properties

The Events Properties screen is the script screen for the area. This screen is shown in Image 3.2.4a below.

Image 3.2.4a – Area Properties Screen – Tab Events



A OnEnter: The OnEnter script will be run each time a **creature object** enters the area. This script will therefore fire when, amongst other times:

- i) a player character enters the area.
- ii) a henchman/familiar/animal companion enters the area.
- iii) a creature is created by an encounter.

iv) a NPC enters the area.

It is therefore important to keep that in mind when you write your script since this script can be engine consuming. Frequently players forget that any creature, not just a player, can cause this event to fire. (The same comment applies to OnExit.) As a result, they may assume in the script body that they are dealing with a PC when they are not. This is a common cause of logic bugs.

- B** OnExit: The OnExit script will be run each time a **creature object** exits the area. This script will therefore fire each time a player character or a henchman/familiar/animal companion exits, by any means or for any reason (including death), the area. The script will also fire when any other creature object exits the area (but not including death).
- C** OnHeartbeat: This script will run every six seconds. It may take more than 6 seconds for the script to run again when there is no player character in the area. In any event, all OnHeartbeat scripts are very engine consuming and should be avoided when possible.
- D** OnUserDefined: This script runs when some specific events are triggered or signalled on the area.
- E** When you click on this arrow, you will have a menu of all available scripts (not only those you have written yourself, but also for all scripts used in the Bioware campaigns).
- F** When you click on these three dots, you will open a window from which you can select any script available for the module.
- G** Clicking on the "Edit" button will launch the Script Editor.

IMPORTANT NOTE: Polymorphing a PC causes them to trigger the OnExit and OnEnter events (in that order), for the area they are currently in.

TIP: If you want the script to be run only when a player character enters the area, then the following script template should be used to save engine power and exit early in the execution of the script when the entering object is not a player:

Script 3.2.4a – Area OnEnter Script Template

```
void main()
{
object oEnter = GetEnteringObject();
if(!GetIsPC(oEnter)) return;
//write below this section your OnEnter script
}
```

TIP: If you want the script to only fire when a player character exits the area, then the following script template should be used to save engine power:

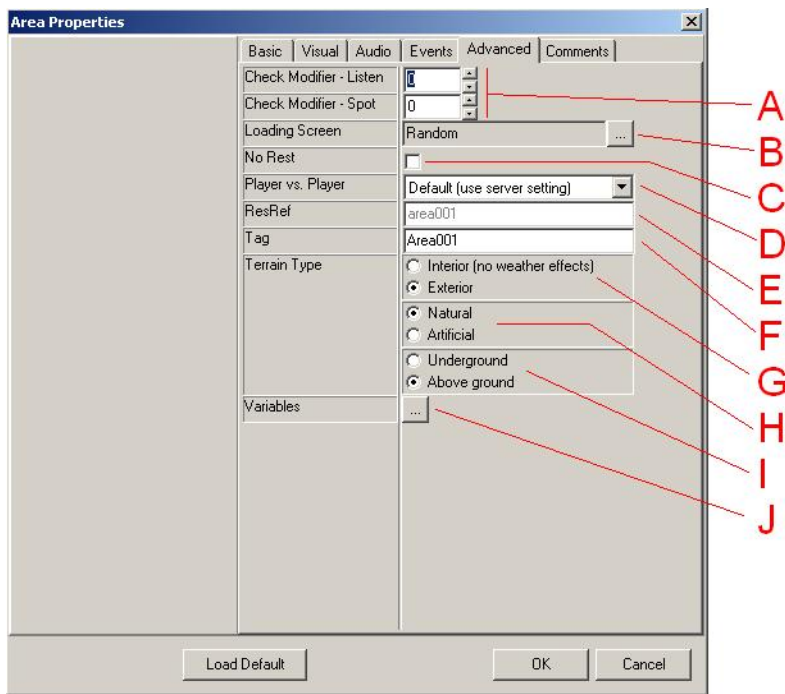
Script 3.2.4a – Area OnExit Script Template

```
void main()
{
object oExit = GetExitingObject();
if(!GetIsPC(oExit)) return;
//write below this section your OnExit script
}
```

3.2.5 The Advanced Area Properties

The advanced properties are the properties of an area that do not fit in any of the other categories, there is nothing advanced about them. Clicking on the Advanced tab will bring you to the following screen:

Image 3.2.5a – Area Properties Screen – Advanced Tab



- A** Check Modifier – Listen/Spot: The Listen and Spot Check modifier can be set between –100 and 100. The values set will be applied on all Listen and Spot checks made in this specific area.

- B** Loading Screen: The Loading Screen is the picture seen by the players while the area is loading when they enter it. By default this loading screen will be set to random and the screen will be chosen between those designed by Bioware for the tile categories applicable to this area (i.e. rural, city exterior, etc.). Clicking on the “...” button will permit you to open the Loading Screen window and, by disabling the “Use Random Loading Screen” box, to choose a specific loading screen that will appear every time this area is loaded.
- C** No Rest: By ticking this box the players will not be able to rest in this area.
- D** Player vs. Player: This property determines if the players are allowed to attack one another in this area. This property can be set to the following values:
- Default (use server setting): means that the player setting the server values in a multiplayer module will set this property for all defaulted areas in the module.
- No PvP: means that a player cannot attack another player.
- Party Protected: means that players within the same party cannot attack each other but can attack players from another party of players.
- Full PvP: means that any player can attack any other player.
- E** ResRef: This is the unique reference to designate the area. This value cannot be changed and does not have any impact on the area itself.
- F** Tag: You can determine or change the Tag of the area in this section. This tag will permit you to designate the area as an object in a script, but otherwise it does not have any impact on any other properties of the area.
- G** Terrain Type: This property determines if the area is an outdoor or indoor area. If this property is set to “Interior”, it will not be possible to set any Wind Power, % Snow, % Rain or % Lightning properties for this specific area (see Sub-section 3.2.2).
- H** Terrain Type: This property determines if the terrain type is natural or artificial. It is not clear if this setting has any impact on any other property of the area. However some skills and feats are influenced by this setting.
- I** Terrain Type: This property determines if the terrain type is underground or above the ground. It is not clear if this setting has any impact on any other property of the area. However some skills and feats are influenced by this setting and it also seems to affect if a player can see past a closed door in some tilesets.
- J** Variables: By clicking on the “...” button you can set local variables on the area. These variables can be used in any script involving this specific area but, otherwise, do not have any impact on any other properties of the area.

3.2.6 The Comments Area Properties

The large empty box appearing when you click on the “Comment Properties” tab is free for you to use at your will. You can keep notes in this area that can be useful for future reference for you or

any other builder working on this module. Adding content in this box will not have any impact on the area or on the module.

3.3 Designing Tips

Do not build overly large areas: Players do not like large areas, unless they serve a specific purpose. Large areas increase the feeling of emptiness, force the player to wander around, and can create lag as well if they are populated by NPCs. Bioware has stated on their public forum that it is strongly recommended not to create areas above 256 tiles (i.e. 16x16, 17x15, etc.). They also suggested that 64 tiles (i.e. 8x8, 7x9, etc.) should be the standard recommended size for an area.

Put borders around your areas: Areas should have borders, except where there is an area transition. This tip only serves logical purposes. If an area does not have a border, players will see an infinite reflection of the ground they are standing on at the end of the area. Therefore the said players will inevitably ask themselves: "I can see that the ground is continuing that way, but why can't I continue to go forward?" It's a lot less frustrating for a player to be stopped by a fence or by a cliff than to be stopped by an invisible and infinite wall.

Build interior areas in accordance with the way they look from outside: A building interior should be generally similar to what the building looks like from the outside.

Do not place a door transition on the very border of an area, it creates an infinite corridor: Infinite corridors occur when you put a door transition on the border of an area. When a player opens the door he will see an infinite corridor instead of a black transition. Fixing this is quite simple, you simply have to resize your area so that the door transition is one tile away from the border of the area.

3.4 How to...

3.4.1 ...copy an area

It might be useful to copy a whole area in situations where, amongst other things:

- i) you want a specific area to evolve at one point in your module (as an example, you want to have the same area before and after a terrible war); or
- ii) you need to script a complicated cutscene taking place in a area and you do not want the other creatures to interfere with your cutscene.

In order to copy a whole area, right-click on the area name in the module structure tree and choose the option "Create Copy". A whole new area will be created, with all the exact same tiles, creatures, doors, encounters, waypoints, etc. You will then be able to do whatever you wish with this "twin" area without affecting the other one. You must however keep in mind that the tag of each object will be exactly the same as the one in the other area. You will have to change these tags to avoid any duplication of objects used in some specific situations. This is quite important with regard to doors and waypoints used in area transitions if you want your area transitions to continue to work properly.

3.4.2 ... modify the properties of a specific tile

Each tile in an area has its own, rather limited, properties. In order to access these properties, you must:

- 1) press the "Select Terrain" button as shown in Image 3.4.2b below;
- 2) right-click on the tile you want to customize; and
- 3) select the tile properties option.

Image 3.4.2a – Tile Properties Screen

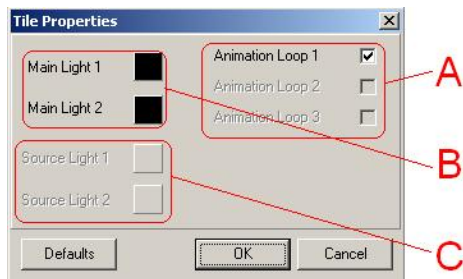


Image 3.4.2b – Select Terrain Button



- A** Animation Loop: The animation loop properties are selectable where there are buildings in your tile that have an animation. An example of this would be the turning windmill in the rural tileset. By ticking the box you will disable any active animation.
- B** Main Light: The main light is basically the light that seems to be coming from above (as if there was a light above in the middle of the tile even if there is none). Setting this to black will cancel this "light from nowhere" effect.
- C** Source Light: Some tiles have independent source lights, like torches on the walls or oil lamps on tables. You can change the color of these independent source lights in this property, or turn them off by choosing the black color.

3.4.3 ...reveal the map of an area to a player

This can be done only through a script. The command to be used to that effect is `ExploreAreaForPlayer()`. As an example, let us say that you want the map of an area to be revealed to a player after the player has asked for directions in a conversation with a NPC. The script that would then be used would be the following:


```
void main()
{
object oPC = GetPCSpeaker();
object oArea = GetObjectByTag("ADD_TAG_OF_AREA");
ExploreAreaForPlayer(oArea, oPC);
}
```

This script would have to be added in the "Action Taken" property of the node of the conversation wherein the player receives information about his whereabouts. (See Sub-Section 5.4.2 about the "Action Taken" properties of a conversation node.)

CHAPTER 4 Creatures

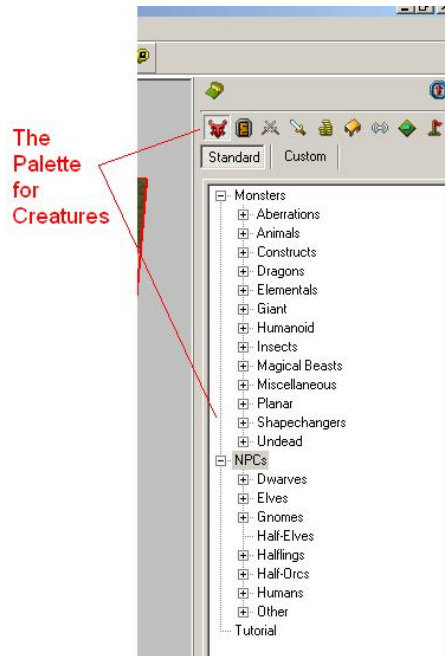
Creatures play an important part of any module. They will act as non player characters, as foes, as henchmen; they will be the living part of your module. There is however no fast way to create a creature. Whether you create it with the Creature Wizard or by using a previously made creature as a template, designing creatures will always require a great deal of property editing to achieve the results you have in mind. This chapter is designed to help you to use the toolset for customizing your creatures.

There are already a multitude of pre-generated creatures that can be chosen from the toolset palette. In order to add one of these creatures to your module you have to choose one in the palette and your mouse will automatically be dragging one of the selected creatures so that you can place it in the area that is opened in the toolset.

The creatures are categorized in the palette by racial type. The Standard palette is composed of the pre-generated creatures made by Bioware. The Custom palette is composed of all creatures you will be making for your module, as well as all custom creatures coming with the Community Expansion Pack (CEP) if you have added this expansion to your module.

The following two sections will show you how to add new creatures to the custom palette. These new creatures are referred to in the toolset as “Creature Blueprint”. These blueprints can be used in encounters (see Chapter 8) or simply as reference to populate your module.

Image 4a – The Palette for Creatures



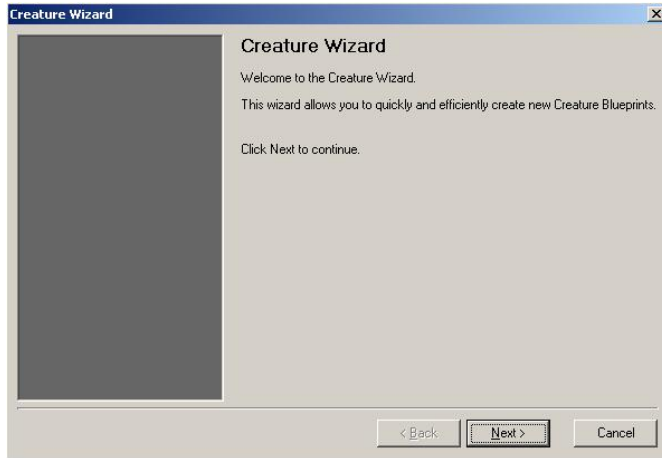
4.1 Creating a Creature Blueprint with the Creature Wizard

There are three ways to launch the Creature Wizard:

- i) by choosing , the option “Creature Wizard” in the “Wizard” menu of the toolset;
- ii) by typing “Ctrl+Alt+C”; or
- iii) by right-clicking on any creature category and selecting the “New” option in the custom palette of creatures.

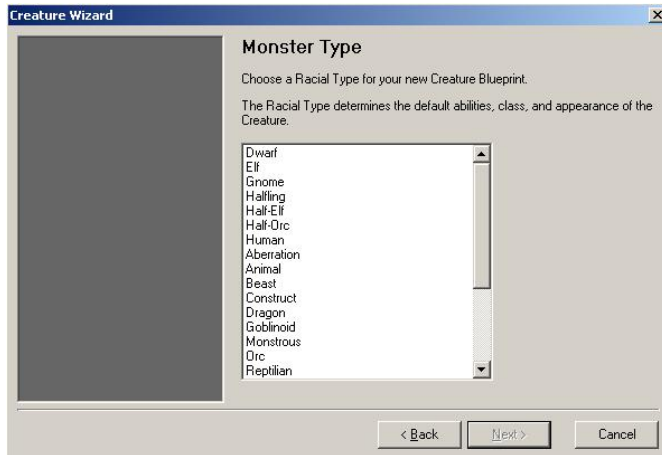
Once the wizard is launched you will get the following screens

Image 4.1a – The Creature Wizard Screen – Step 1



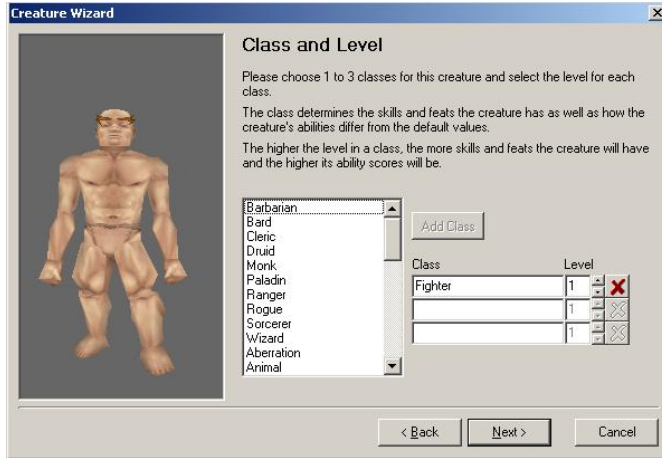
STEP 1: This screen is simply a warm welcome and there is nothing else to do at this stage but to press the "Next >" button (or on the "Cancel" button if you already regret launching the wizard).

Image 4.1b – The Creature Wizard Screen – Step 2



STEP 2: In this second screen, you must select the racial type of the creature you wish to create. As mentioned in this screen, the racial type will determine the default abilities, class and appearance of the creature. These properties can be changed at a later time.

Image 4.1c – The Creature Wizard Screen – Step 3



STEP 3: The information appearing on this screen is self-explanatory. Please note however that the skills, feats and ability score that will be given to the creature for any given attributed level of the creature will be the default ones. As a result, all fighters created with the wizard will be specialized in longsword and will allocate all their extra ability scores to strength, all sorcerers will choose the exact same spells, etc.

As a result, it is impossible to add prestige classes during this process.

Fortunately, it is possible to change all the default allocated skills, feats and ability scores as well as adding new levels to the creature (including in any prestige classes) at a later point.

Image 4.1d – The Creature Wizard Screen – Step 4



STEP 4: In this screen you can select the Gender, the Appearance and the Portrait of your creature. The appearance will not modify the racial type or the default abilities assigned to the creature. You can therefore create a gnome bard with the appearance of a troll, but in the end your creature will have the attributes of a gnome bard. These properties can be modified later on.

Image 4.1e – The Creature Wizard Screen – Step 5



STEP 5: In this screen you determine the faction of your creature. Factions are explained in another section this chapter (see Section 4.4 below). Again, this property can be modified later on.

Image 4.1f – The Creature Wizard Screen – Step 6



STEP 6: This step requires you to choose a name for the creature you are creating. If the creature is of one of the standard NPC racial types (dwarf, elf, gnome, half-elf, halfling, half-

orc or human), you can use the random name generator if you lack inspiration (this can be done by clicking on the “Random” button). Again, this property can be changed at a later stage.

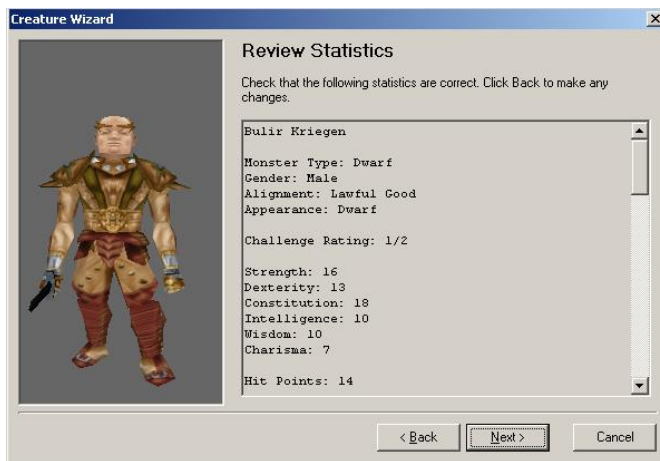
Image 4.1g – The Creature Wizard Screen – Step 7



STEP 7: Here you need to select in which palette category you want the newly made creature to appear. These palette categories are organized by racial types, but nothing prevents you from creating the blueprint in a category that does not describe your creature.

If you use the Community Expansion Pack and you do not want your own custom creatures to be mixed up with those coming with this pack, then it might be a good idea to use the “Special” category to organize your custom creatures for easier reference when you build your module.

Image 4.1h – The Creature Wizard Screen – Step 8



STEP 8: Here you have a summary of the creature you created. This is for review only and no changes can be made on this screen.

Although it is allowed to click “Back” to make any changes, you will not have the opportunity to change the default settings of the creature by doing so. Clicking back will simply bring you back to the previous 7 steps. If you want to modify any property of the creature it is preferable click “Next” instead of going back and to launch the Creature Property Window as described in Step 9.

Image 4.1i – The Creature Wizard Screen – Step 9



STEP 9: Your new creature is now created. It will appear in the custom creature palette in the toolset window in the category you selected in Step 7. You may select the “Launch Creature Properties” box if you want to open the Creature Properties Screen at this stage to further customize this creature. (See Section 4.3 for more details on how to change these properties.)

4.2 Creating a Creature Blueprint from an Existing Creature

Another way to create a new creature is by creating a copy of an existing one and customizing it to fit your purposes.

In order to do so, right-click on the creature in the palette you want to use as a model, and select the “Edit Copy” option in the dropdown menu. This causes the Creature Properties Screen to open so that you can customize this creature at will.

Another way to create a creature from an existing one is to place a creature of your choice in an area and edit its properties. Once you are finished, right-click on this property and select the “Add To Palette” option to make it a blueprint in the palette. The major problem caused by using this method is that the toolset will try to make custom blueprints for every item your creature is carrying (even if these items are not custom items) and you will see a warning screen to that effect when you chose the “Add To Palette” option. In order to avoid this, you must click on the “X” button on every pop-up item blueprint screen that appears on your screen, until the creature blueprint screen

appears. When the creature blueprint pop-up screen appears, click the “OK” button to add the creature to the creature palette.

IMPORTANT NOTE: It will always be possible for you to make modifications to a creature blueprint at a later time. Do not therefore feel that your creature needs to be perfect before you add it as a blueprint to the palette.

4.3 The Properties of a Creature

In this section of the guide we will provide you with detailed comments on each of the properties that can be customized when you open the Creature Properties screen. You can access this screen in many ways.

If you want to change the properties of a specific creature placed in one of your areas, you can:

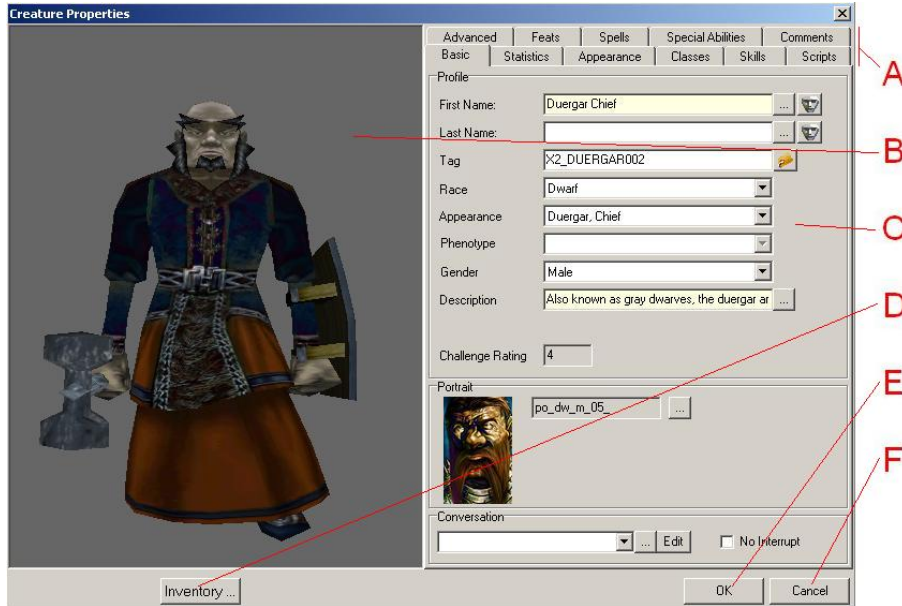
- i) Right-click on the creature and select the “Properties” option; or
- ii) Select the creature and, in the “Edit” menu of the toolset, select the “Creature Properties” option.

If you want to change the properties of a creature blueprint in the Custom creatures palette, right-click on this creature and select the “Edit” option in the menu that appears.

There is no way to change the properties of a creature blueprint in the Standard Creatures palette. These creatures were designed by Bioware and used in the original campaigns. You must therefore create a new creature blueprint from these standard creatures (see Section 4.2 above) before you can make any modification to them.

Once you open the Creature Properties screen, you will get the following window:

Image 4.3a – The Creature Properties Screen



A As you see, the properties of the selected creature are grouped into 11 categories represented by the tabs at the top of the screen, namely:

- i) the basic properties (see Sub-Section 4.3.1);
- ii) the statistics of the creature (see Sub-Section 4.3.2);
- iii) the appearance of the creature (see Sub-Section 4.3.3);
- iv) the classes of the creature (see Sub-Section 4.3.4);
- v) the skills of the creature (see Sub-Section 4.3.5);
- vi) the scripts attached to the creature (see Sub-Section 4.3.6);
- vii) the advanced properties (see Sub-Section 4.3.7);
- viii) the feats of the creature (see Sub-Section 4.3.8);
- ix) the spells of the creature (see Sub-Section 4.3.9);
- x) the special abilities of the creature (see Sub-Section 4.3.10);
- xi) the comments related to the creature (see Sub-Section 4.3.11);

Each of these categories of properties will be described in greater detail below.

B In this screen you can see how your creature will look in the game.

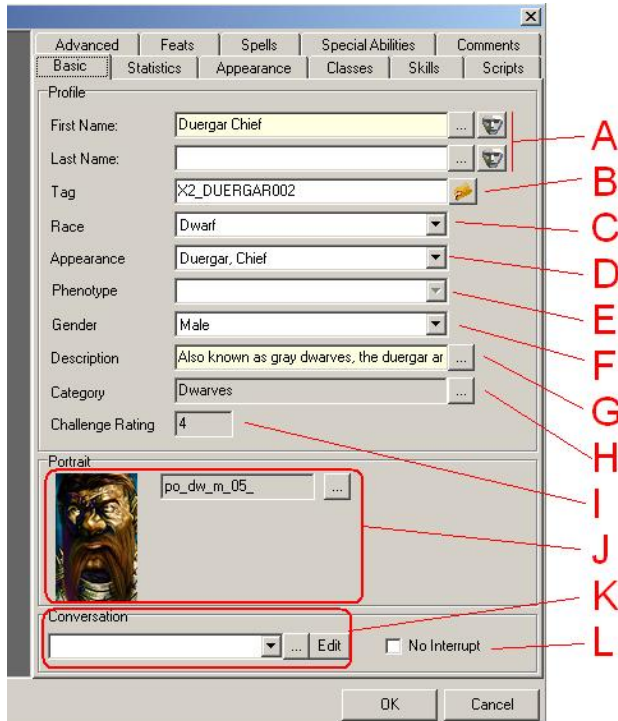
C This is the property screen. You will find in this screen all the properties of your creature, organized in the categories referred to by the tabs mentioned in **A**.

- D Clicking on the “Inventory” button will open the Inventory Contents window, which is more fully explained and described in Sub-Section 4.3.12 below.
- E If you click on the “OK” button you will keep all the modifications you have made to this creature’s properties since opening the Creature Properties Screen.
- F If you click on the “Cancel” button you erase all the modifications you have made to this creature’s properties since opening the Creature Properties Screen.

4.3.1 The Basic Properties of a Creature

Clicking on the “Basic” tab of the creature properties will bring you to the following screen:

Image 4.3.1a – The Creature Properties Screen – The Basic Tab



- A First / Last Name: Here you can edit the names of the creature.

It is important to keep in mind that, in the game, only the first name will appear to the player when they scroll their mouse over the creature or press on the “Tab” button. Therefore, if you want players to see the full name of your creature on mouse scroll, you must put both the first and last name of the creature in its first name property.

If you click on the “...” button, you will open the “Edit String” screen which permits you to edit the name for each one of the international versions of NWN. This is useful only if you intend to build a module that will be available in many different languages.

If you click on the button showing a little face, you will generate a random name for the creature. This possibility is available only if the creature is of one of the standard NPC racial types (dwarf, elf, gnome, half-elf, halfling, half-orc or human).

- B** Tag: The tag of a creature permits you to designate it in a script or in a multiparty conversation.

Unless you need to identify this creature in a specific script or in a multiparty conversation, there is no need to change the default tag.

Otherwise, it is highly desirable to assign a unique tag to your creature. It is a good idea to establish a naming convention for the module you are building before assigning a unique tag. This will permit you or any other builder working on your module to avoid duplicating tags, and it will also be easier for you when you are writing scripts to remember the tags of your creatures. Naming conventions are often based on the tag of the area in which the creatures are located.

If you do not want to take the risk of duplicating tags that need to be unique, you can click on the button next to the tag property (showing a tag icon) and the toolset will assign a unique tag to your creature.

- C** Race: This is the race of your creature. Changing the race will NOT change the appearance of your creature, nor will it adjust the feats or skills assigned to the creature that were specific to its racial type at creation. Changing the race will however change the “Racial Modifier” affecting the ability scores of the creature. (See Sub-section 4.3.2.)

Although changing the race will not heavily modify the creature, it is important to keep in mind that many items have racial based characteristics (examples: AC bonus vs. racial group, Attack bonus vs. racial group, etc.), which of course will take this property into consideration.

- D** Appearance: The appearance of a creature determines how the creature will look in the game.

Changing this property will also have minor impacts on other characteristics of the creature, namely:

- i) the armor class size modifier (see sub-section 4.3.2 below);
- ii) the ability of the creature to wield large weapons.
- iii) the default movement rate speed of the creature (see sub-section 4.3.2 below).

On this subject, it is important to note that changing the appearance of a creature (whether in the toolset or by a script) will automatically reset the creature speed to its default one.

- E** Phenotype: This characteristic is purely esthetical and is available only when the creature appearance is set to one of the standard NPC racial types (dwarf, elf, gnome, half-elf, halfling, half-orc or human). By this property, you can determine if the creature is of a normal shape or of a large shape. Changing this characteristic will have no effect at all on the creature but to change the way it looks. The creature can still wear any armor it could before the change even if it is of a larger phenotype.
- F** Gender: Here you can set the gender of the creature. The gender can be set to “Male”, “Female”, “Both”, “Other” or “None”. Changing the gender from male to female will change the appearance of the creature only if the creature appearance is set to one of the standard NPC racial types (dwarf, elf, gnome, half-elf, halfling, half-orc or human). If the creature appearance is of one of these standard NPC racial types, setting the gender to “Both”, “Other” or “None” will assign the male appearance to the creature.
- G** Description: All creatures pre-generated by Bioware come with a description. If you create your own creatures with the wizard, then you will have to write your own descriptions.
- The description will appear to the player when they examine the creature. It is important to know that many players examine creatures to see their challenge rating, so they will notice if you do not add any description to your creatures.
- The “...” button next to the description may be used to open the “Edit String” window to write descriptions in the other languages supported by NWN. This is useful only if you intend to make your module available in multiple languages.
- H** Category: The “Category” property informs you where in the palette the blueprint of this creature is located. Note that this property only appears when you are editing a blueprint. You will not see it if you are editing a placed creature.
- You can move your blueprint from one category to another by clicking on the “...” button. If you use the Community Expansion Pack and you do not want your own custom creatures to be mixed up with those coming with this pack, then it might be a good idea to use the “Special” category to organize your custom creatures for easier reference when you build your module.
- I** Challenge Rating: The challenge rating is a rate given to the creature by the toolset. This rate is established in consideration of the abilities, levels and inventory of the creature. The challenge rating is used in calculating encounters (see Chapter 10). It determines how many creatures will be spawned to face the player characters, depending on the difficulty set for the encounter. The challenge rating may be adjusted in the “Advanced” properties tab. (See Sub-section 4.3.7.) The challenge rating is also one of the factors used by the game to determine how many experience points will be given to a player character that kills this creature.
- J** Portrait: You can change the portrait of your creature by clicking on the portrait itself or on the “...” button.
- K** Conversation: Here you can assign a conversation file to your creature. Clicking on the arrow will show you a dropdown list of all conversation files from the Bioware official campaign and those specific to your module. Clicking on the “...” will open a window in which you will see all conversation files that you have created. Clicking on the “Edit” button will

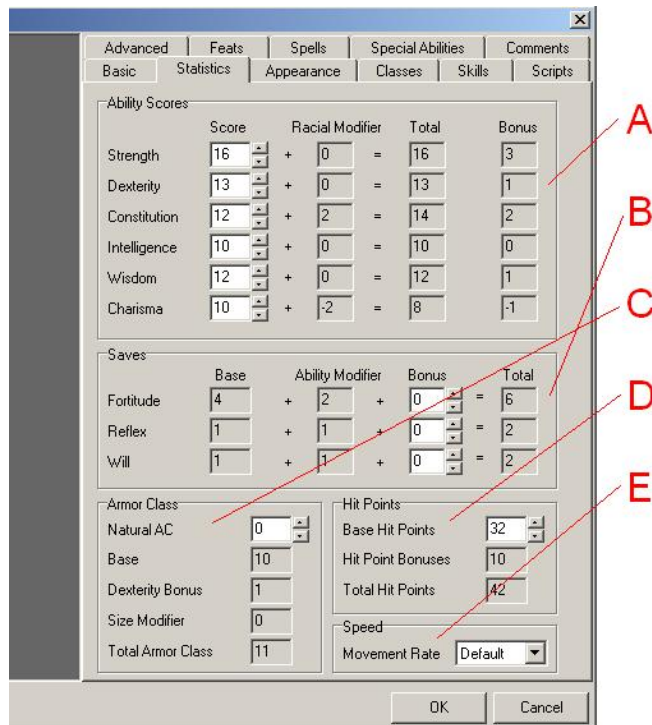
launch the Conversation Editor. (See Chapter 5 for detailed comments on how to create a conversation and use the Conversation Editor.)

- L** No Interrupt: Ticking this box will make it impossible for a player to interrupt the conversation this creature is having with another player. If your module is intended to be playable in multiplayer mode, it might be advisable to tick this box on most important NPCs because there is nothing more annoying than seeing a fellow player interrupting your conversation.

4.3.2 The Statistics Properties of a Creature

This screen will provide you with the details of the numerical abilities of the creature.

Image 4.3.2a – The Creature Properties Screen – The Statistics Tab



- A** Ability Score: Here you have the common ability scores of your creature. The default scores were allocated at the creation of the creature in accordance with the default level up option for the classes you chose for the creature. You can however modify these scores at will.

The Racial Modifier will change depending on the racial type of the creature. The racial type can be changed in the “Basic” tab.

The Bonus is determined depending on the total ability scores. These bonuses will affect many dice rolls made in the game.

- B** Saves: The “Saves” statistics determine how well the creature will defend itself when a spell or a special attack is attempted on them.
- C** Armor Class: The “Armor Class” of the creature will determine how well the creature will resist physical attacks from its opponents. The only statistic that can be modified in this area is the “Natural AC”. Unfortunately only positive scores can be added to this property and therefore you cannot use it to reduce a creature’s AC.

WARNING: The statistic appearing in the creature property does not take into consideration the armor or magical items currently worn by the creature. Therefore, before considering raising the natural AC of the creature you have to determine if the armor worn by the creature is a sufficient defence.

- D** Hit Points: The “Base Hit Points” has been set by the toolset by using default parameters during the creature creation process. Hit points will be raised if you add classes to the creature by using the “Levelup Wizard”. (See Sub-section 4.3.4.)

The “Hit Point Bonuses” are determined by Constitution Bonus and the number of class levels given to the creature.

You can modify the base hit points statistic at will. Keep in mind that, by doing so, it will have a direct impact on the creature challenge rating.

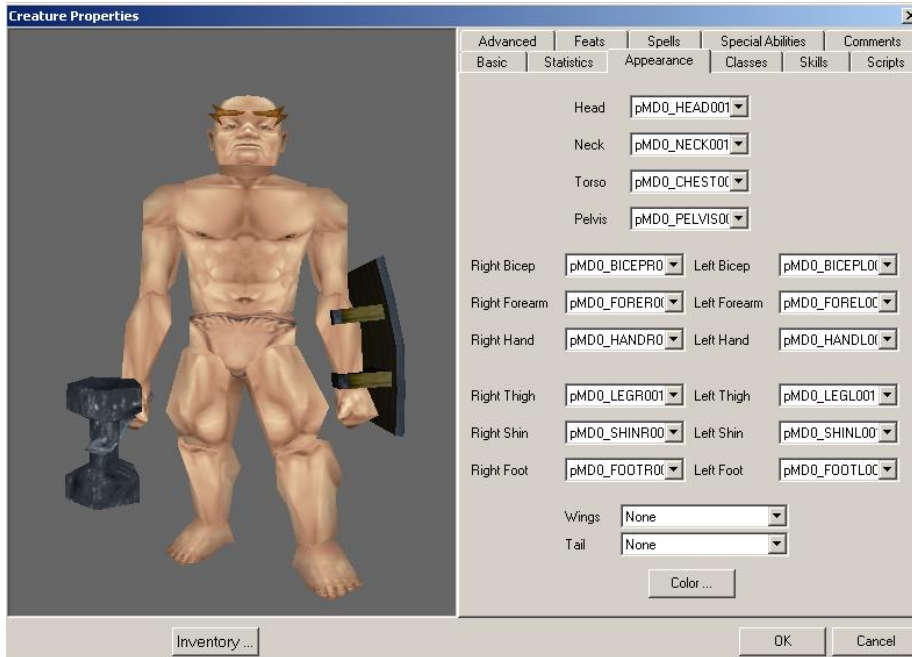
- E** Speed: The default movement rate is set depending on the creature appearance. A creature having the appearance of a zombie or a mummy will be very slow. You have here the opportunity to change the creature’s speed. Keep in mind however that if you change the Appearance property of the creature (whether by modifying it under the “Basic” tab or through a script) the movement rate speed will be set back to its default value.

4.3.3 The Appearance Properties of a Creature

Do not be too impressed by all the information appearing in this screen. The appearance window is unfortunately not as customisable as we may think. First of all, only the standard NPC racial types (dwarf, elf, gnome, half-elf, halfling, half-orc or human) can be modified in this window, except for a few other creatures to which you can add wings or a tail through this properties screen. And even with the standard NPC racial types, the only feature that provides you with a lot of choice is the head of the creature. For all other parts of the creature (except the wings and tail), you only have three choices: (1) no part at all; (2) the standard part; or (3) the tattooed part.

Here is the creature appearance screen:

Image 4.3.3a – The Creature Properties Screen – The Appearance Tab



The color of the skin, the hair and the tattoos can be modified by clicking on the “Color...” button.

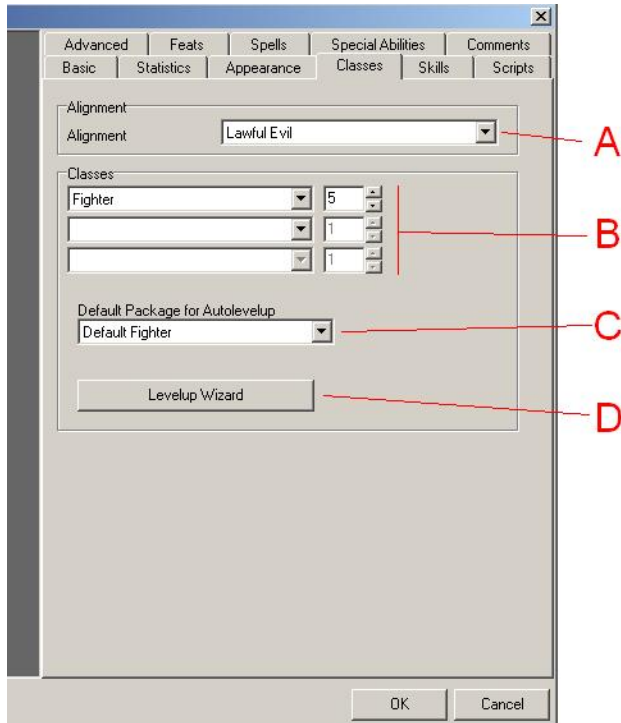
Modifying the appearance properties in this screen will have no impact at all on the other properties of the creature.

TIP: There are some hidden colors in the character color palette. These colors are true black, true white and many cool metallic colors (to be used in moderation, unless of course you want to create a tribe of dwarven robots). To obtain these hidden colors, simply press on any color of the palette and hold down your mouse button, then move the mouse down in the lower part of the color palette window where there is no color, and finally let your mouse button go.

4.3.4 The Classes Properties of a Creature

Clicking on the “Classes” tab will bring you to the following screen:

Image 4.3.4a – The Creature Properties Screen – The Classes Tab



- A** Alignment: Alignment does not have much to do with classes, but they have included this property on this screen. This property does not have much impact on the creature. The only one worth mentioning is that many items in the game are alignment restricted or have some bonuses vs. some specific alignments.
- B** Classes: Here you will see the number of levels per class given to the creature. Like player characters, a creature has a maximum of three different classes.

WARNING: Changing the classes directly in this screen will not adjust the feats, skills, spells or any other ability that is gained by levelling up. Changing the classes and level allocated to each class here will only affect abilities that are based on the number of levels that a creature has.

Example: If you raise this creature from a level 5 fighter to level 10, it will adjust the hit points bonuses accordingly (since this statistic is based on the following formula: Constitution Bonus X # of levels) but will not raise the Base Hit Points statistic since base hit points are allocated in the levelling up process and not according to some formula.

If you want the creature to properly level up and have default feats, skills, spells, etc. assigned to it in the levelling up process you must use the "Levelup Wizard". (See **D** below.)

- C Default Package for Autolevelup: This parameter will only determine what class will be used by default when levelling up through scripting.

To level up with a script, you must use the LevelUpHenchman() command (even if the creature is not an henchman).

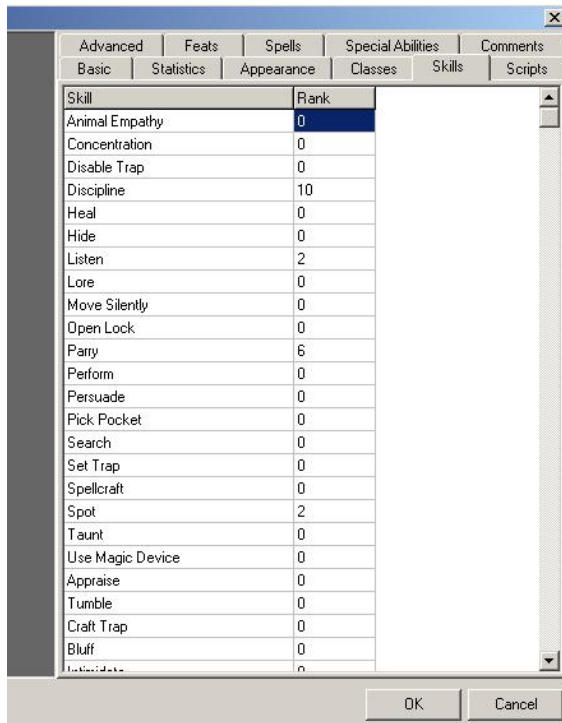
You must be aware that in order to level up a creature through a LevelUpHenchman() command, that creature must have followed its packages stringently. Therefore, if you have added, as an example, feats manually to your creature, then you will no longer be able to use the LevelUpHenchman() command successfully on that creature.

- D Levelup Wizard Button: The Levelup wizard can be used to add class levels to a creature. By using this wizard, the toolset will assign the default feats, skills, stats, abilities and spells that are designated to the classes you choose to level up in. You can level up many levels at once by using this wizard.

4.3.5 The Skills Properties of a Creature

In this screen you will be able to see and modify the skills of your creature.

Image 4.3.5a – The Creature Properties Screen – The Skills Tab



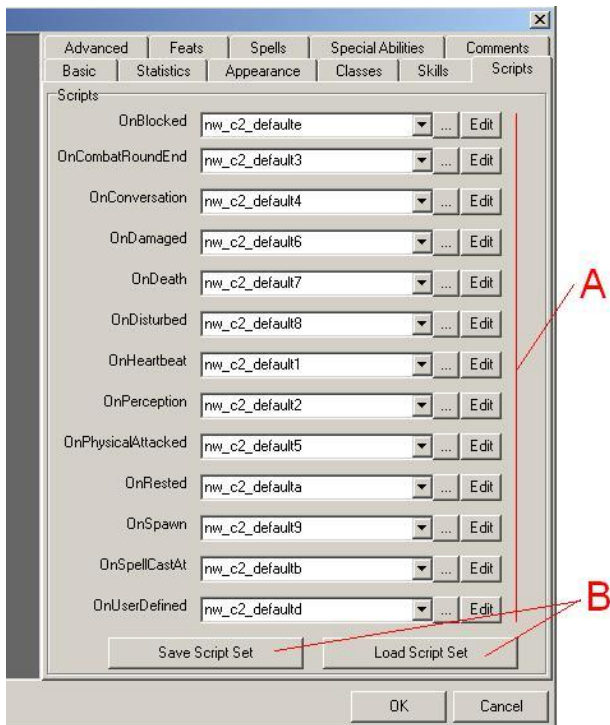
Most of these skills do not have any impact on the game unless the creature is used by players as an henchman. Otherwise, the only skills that are worth adjusting are the following ones:

- i) the “Concentration” skill is really important for creatures that are intended to be used as enemy spell casters. This skill is used to determine if a creature is able to cast a spell while it is being attacked by another character.
- ii) the “Listen” and “Spot” skills may have an impact on the ability for enemy creatures to notice player characters that are hidden in the shadows.
- iii) the “Tumble” skill will determine if opponents will have an attack of opportunity on this creature if it is leaving close combat.

4.3.6 The Scripts Properties of a Creature

In the following screen you can customize the default scripts of your creature:

Image 4.3.6a – The Creature Properties Screen – The Scripts Tab



There are two sets of default scripts for creatures that are not meant to be henchmen. The scripts appearing in Image 4.3.6a are the ones coming with the original NWN game and you will find them on almost all pre-generated creatures made by Bioware. Since HotU came out, these scripts are no longer used when you make a creature with the creature wizard, you now have scripts starting by

"x2_def_***" instead. In the end, these default scripts are almost identical so beginner and intermediate builders should not concern themselves with which default script set is assigned to their creatures.

These scripts may be changed to include customized behaviour or events for the specific creature you are working on. Each script is assigned to a particular event. The Lexicon provides the following explanation for each of these events:

OnBlocked: The script attached to this event fires when a creature is blocked by a door. It can be used for a dramatic door crashing down scene or to cause the creature to stop pursuit of anything it was chasing.

OnCombatRoundEnd: The script attached to this event fires at the end of each combat round (every six seconds in NWN). It will fire every 3 seconds for a hasted creature.

OnConversation: The script attached to this event fires whenever the creature is clicked upon by a non-hostile player character or when a shout is heard.

By default, if the creature has been clicked on for conversation, it will stop what it is doing and begin dialogue (if it can). Otherwise the script checks to see if a spoken (sometimes typed) phrase is recognizable and if the creature has been set up to recognize combat shouts. Only if the creature is listening (`SetListening()`) will this event fire at all. Once the creature is set to listening, they will only hear things set by `SetListeningPattern()`. The script "nw_g0_conversat" will fire if there is no `OnConversation` script supplied, and will start the creature's conversation if someone clicks on them.

OnDamaged: The script attached to this event fires when the creature is damaged. The default script provides that, if a creature was hurt by someone they cannot see, they attempt to search for the attacker. If they can see their attacker, they locate and attack it.

OnDeath: The script attached to this event fires when the object dies for any reason. It can be used to force the object to drop treasure, respawn at a new location, reincarnate, or cause a creature to spawn.

If a creature dies, the default `OnDeath` script checks to see if the creature was non-evil and had Commoner levels - if so it automatically changes the killer's alignment a little towards evil. It also emits standard shouts for help to any creatures within range who are set up to listen for them. Right after this script is fired, the last killer is awarded experience from the hardcoded Bioware system (even if set to 0%).

`DestroyObject` will not trigger this event, although DM kill will (with an invalid last killer).

Scripts that execute in this event should have their actions take effect immediately. Using `DelayCommand` will work only so long as the corpse of the creature remains. Although `AssignCommand` could be used to place an instruction in the action queue of another object, for example an invisible object placeable or the module itself. Also note that if `DestroyObject(OBJECT_SELF)` is used in the `OnDeath` event, it will cause associated `UserDefined` events, ie the `Death` event, to not execute at all.

OnDisturbed: The script attached to this event fires when the inventory of the creature is changed. This will generally occur on a pickpocket attempt since a NPC creature cannot otherwise have an item added to or removed from its inventory (it is not a container).

OnHeartbeat: The script associated with this event fires every six seconds. This is good for a script that needs to check for something repeatedly or perform an action regularly. Be aware that too many OnHeartbeat scripts can be processor costly. Use other types of event scripts when possible.

OnPerception: The script attached to this event will fire whenever the creature perceives another creature or a player character. Common uses of this script include having the creature yell a battle cry, greet the player, or run away.

The OnPerception default script gives priority to first going into search mode if an enemy has suddenly vanished, then telling the creature to attack an enemy that has appeared.

OnPhysicalAttacked: The script attached to this event fires each time the creature takes physical damage. This could be used for a reaction to the attack such as changing weapons, fleeing, surrendering, etc.

If a creature is attacked while it is not in combat mode, it goes hostile. The default script provides that the creature emits appropriate shouts to alert nearby allies. It should also be possible to use this routine to allow a member of a non hostile faction to issue a warning to a PC that attacked them, going hostile only on a second attack. At least that seems to be the purpose of the 'NW_FLAG_SET_WARNINGS' flag found in the on spawn routine. However, because the code that makes the faction and allied faction members hostile to the offending player is hard-coded into the engine itself, it is impossible to know for sure why a particular creature has become hostile. As such there seems to be no easy way to actually get this functionality working.

OnRested: The script attached to this event fires when the creature attempts to rest. It can be used to customize the behaviour of the creature so that instead of just sitting down it could drink from a tankard or dance around to regain lost hit points. However, creatures do not normally rest so the default script is empty.

OnSpawn: The script attached to this event fires when the creature is first spawned (i.e. created in the module) and allows you to specify the default behaviours for the creature (examples: does the creature flee when attacked, walk a set of waypoints, etc).

OnSpellCastAt: The script attached to this event fires when the object becomes the target of a spell. This can include responses such as casting a spell of the object's own, dying, disappearing, getting stronger (absorbing the magical effect), etc.

OnUserDefined: The script attached to this event fires depending on settings made during the OnSpawn event for creatures and some general level blanket signals. Typically this script is a switch/case statement that does something different depending on the signal (numerical value) sent to it by an event that calls the script. This event makes it very easy to retain the default behaviour of an object while adding custom mannerisms of your own.

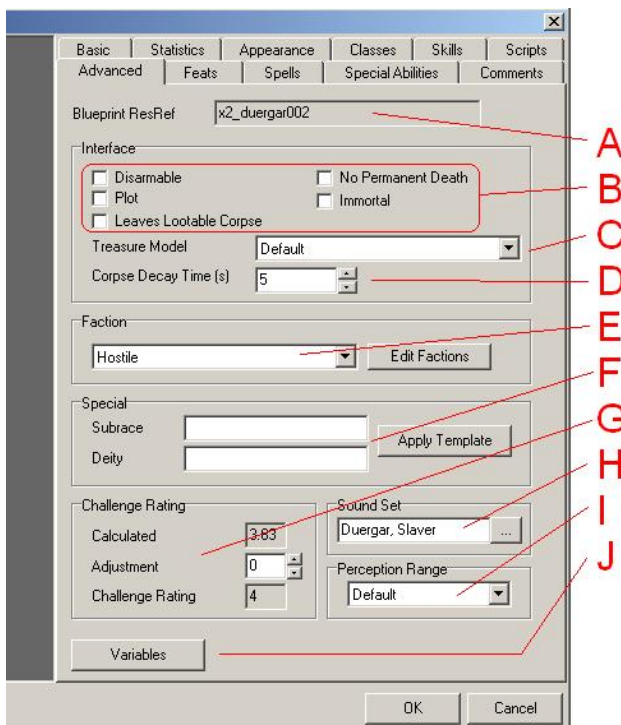
WARNING: when you edit one of the default scripts, do not forget to change its name or otherwise all the creatures in your module using the default script will now run the modified script you have made.

If you know your way around in NWN scripting, note that it is possible to set up the character so no changes are made to any of the default handler scripts except the OnUserDefined script, which has been set aside for custom use. The Variables section of the character can contain special values to cause either a PreSpawn (EVENT_USER_DEFINED_PRESPAWN) or PostSpawn (EVENT_USER_DEFINED_POSTSPAWN) event to be passed to the OnUserDefined script, and all other event scripts have code to cause an associated UserDefined event if it has been activated of that event. In this way, the character can take an event at Spawn time to activate any other events pertinent to this character, and handle them in the appropriate case statement of the switch statement. Since this leaves the default scripts intact, and collects all the character-related event handling in one place, it represents a very desirable way to handle events for characters.

4.3.7 The Advanced Properties of a Creature

As for any object properties, the Advanced tab is simply the gathering of the properties that do not fit in any of the other categories, there is nothing really advanced about them. By selecting the Advanced tab, you will see the following screen:

Image 4.3.7a – The Creature Properties Screen – The Advanced Tab



A Blueprint ResRef: ResRef is a unique reference name differentiating objects in the palette.

The ResRef permits the game to determine from which blueprint an object placed in an area was created. There cannot be two objects in the palette with the same ResRef. This is however not a concern for you since the toolset will automatically assign to any new blueprint a new and unique ResRef. This “ResRef” property is primarily useful when:

- i) you want to create a creature through a script; or
- ii) you want to update all the creatures having the same ResRef that you have placed into your module or into a specific area so that the modifications you have made to the blueprint of the creature be made to the placed creatures as well. This can be done by right-clicking on the creature blueprint in the creature palette and choosing the “Update Instances” option in the dropdown menu.

The ResRef may only be modified at the moment the creature is created. After the creation process, this reference becomes permanent.

B These properties, when enabled, have the following effects:

Disarmable: determines if the creature can be disarmed in combat.

Plot: if the creature is designated as a plot object, then it will not be possible to damage or kill it. Furthermore, the creature will not turn hostile when attacked by a player. This property is generally used to ensure that a main quest related NPC (like Aribeth in the original campaign) is not affected by any hostile action so that the plot cannot be broken by mistake or by a player acting stupidly in a module.

This feature can also be turned on and off by a script using the SetPlotFlag() command.

Leaves Lootable Corpse: This determines if the creature’s corpse will be left on the ground when the creature is killed instead of leaving a treasure model. The corpse will however be left only if the creature has droppable inventory that can be looted by the players. Once the corpse is looted, then the corpse will fade after the allocated time period specified for that effect in the “Corpse Decay Time(s)” property.

If you want the corpse of the creature to stay permanently on the ground, even if the corpse is not lootable or has already been looted, then you must add a SetIsDestroyable() command in the OnSpawn script of the creature.

No Permanent Death: When the creature suffers a lot of damages from the blow that kills it, there might be an animation making the corpse explode. If you do not want the corpse of the creature to explode at any time, then you must tick this box.

Immortal: When a creature is set immortal, it will suffer normal damages and will turn hostile when attacked (which is not the case for a plot creature as mentioned above), but the creature will never die.

This feature is useful if you want the creature to surrender at some point and avoid the creature being killed by a critical blow before it has the chance to surrender.

This feature can also be turned on and off by a script by using the SetImmortal() command.

- C** Treasure Model: If you have not enabled the “Leaves Lootable Corpse” property, the creature upon death will leave a treasure model on the ground, as long as the creature has droppable items or a random treasure script. If you use the default OnSpawn script for your creature, there will automatically be a random treasure left on the ground upon death of the creature. The “Treasure Model” determines what this treasure will look like for the players. The default model will generally be set depending on the creature’s appearance (a pouch for most creatures and bones for skeletons, as examples). You can modify the default treasure model to any other one in the list provided for this property.
- D** Corpse Decay Time(s): Here you can set how many seconds the corpse will be left on the ground after it has been looted (if the killed creature had any loot) or, otherwise, after death. This will only work if you have enabled the “Leaves Lootable Corpse” property. This property can be set between 5 and 32767 seconds.
- E** Faction: Here you can set the creature’s faction. (See Section 4.4 for more details on this feature.)
- F** Subrace / Deity: You can set the subrace and deity of your creature here. For these properties to have an impact on your creature you will need fairly advanced knowledge of 2da editing and scripting.
- G** Challenge Rating: The challenge rating is a rate given to the creature by the toolset. This rate is established based on the abilities, level and inventory of the creature. The challenge rating is used in calculating the encounters (see Chapter 10), to determine how many creatures will be spawned to face the player characters depending on the difficulty set for the encounter. The challenge rating may be adjusted here if you consider that the rating established by the toolset creates balance issues in your encounters. Note that the challenge rating is also one of the factors used by the game to determine how many experience points will be given to a player character that kills this creature.
- H** Sound Set: By clicking on the “...” button you will open a window in which you can hear and select the sound sets that are available to the creature. The sound set will establish the sound the player will hear when the creature is hit, when the creature is attacking, or when the creature dies. Some voice sets (mainly those of the standard NPC racial types) will also have greetings and some other social sounds included.
- I** Perception Range: The perception range establishes from how far away a creature can perceive another creature. When another creature enters in the perception range of the creature, the OnPerceived script event will run, making a hostile creature attack the perceived creature.
- J** Variables: By clicking on the “Variable” button you can set local variables on the creature. These variables can be used in any script involving this specific creature.

Some specific variables have already been coded by Bioware and can be used to modify the behaviour of your creatures. Here is a list of these already coded variables:

- i) X2_SPELL_RANDOM

Setting this variable to 1 on a spellcaster creature will make its spell use a bit more random, but their spell selection may not always be appropriate to the situation anymore.

- ii) X2_L_SPAWN_USE_STEALTH
Set to 1 to make the creature activate stealth mode after spawn.
- iii) X2_L_SPAWN_USE_SEARCH
Set to 1 to make the creature activate detect mode after spawn.
- iv) X2_L_SPAWN_USE_AMBIENT
Set to 1 to make the creature play mobile ambient animations after spawn. The creature will then move in the area where it is located, salute other creatures, scratch its head, etc.
- v) X2_L_SPAWN_USE_AMBIENT_IMMOBILE
Set to 1 to make the creature play immobile ambient animations after spawn.
- vi) X1_L_IMMUNE_TO_DISPEL
Set to 1 to make the creature immune to dispel magic (used for statues).
- vii) X2_L_IS_INCORPOREAL
Set this variable to 1 on a creature to make it walk through other creatures
- viii) X2_L_NUMBER_OF_ATTACKS
Set this variable to 1 - 6 to override the number of attacks a creature has based on its base attack bonus.
- ix) X2_L_BEH_MAGIC
The value of this variable (int) is added to the chance that a creature will use magic in combat. Set to 100 for always, 0 for never.
- x) X2_L_BEH_OFFENSE
The higher value of this variable, the higher the chance that the creature will use offensive abilities in combat. Set to 0 to make them flee.
- xi) X2_L_BEH_COMPASSION
The higher value of this variable, the higher the chance that the creature will aid friendly creatures in combat. Note that helping usually degrades the overall difficulty of an encounter, but makes it more interesting.
- xii) X2_S_PM_SPECIAL_ITEM

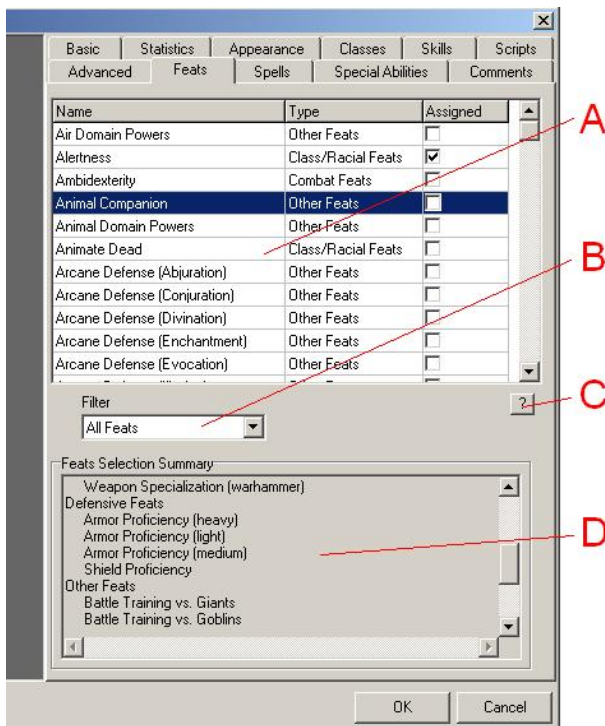
This allows you to script items that enhance a palemaster's summoned creatures. You need to put the name of a script into this variable that will be run on any creature called by the pale master's summon undead ability. You can use this script to add effects to the creature. You can use the OnEquip/OnUnEquip event hooks set this variable.

4.3.8 The Feats Properties of a Creature

If the Creature Wizard was used to create this creature, the feats appearing in this screen will be those assigned by default by the toolset to the creature depending on its racial type and classes. It will therefore be important to adjust these feats so that the creature is proficient for the situation in which the creature will take part. Especially for combat, you will notice that the default feats are always aimed towards some very specific weapons (mainly the longsword) and will have to be adjusted if you want your creature to wield other weapon types.

There is no limit to the number of feats which may be assigned to the creature. These characteristics are therefore highly customizable.

Image 4.3.8a – The Creature Properties Screen – The Feats Tab



- A** In this window you have the list of all available feats. Ticking the “Assigned” box will assign the associated feat to your creature.

- B** The list of feats is very long and you may not want to see all of them when you customize your creature. By using the filter you can choose to only see Active Combat Feats, Class/Racial Feats, Combat Feats, Defensive Feats, Magical Feats and Other Feats.
- C** By clicking on the “?” button you will open a small window in which will be the description of the specific feat or feat category (when there is more than one level to a feat) that is selected.
- D** In the “Feats Selection Summary” you can see what feats were given to this creature. This list is adjustable by assigning or removing feats in the feats window (see **A**).

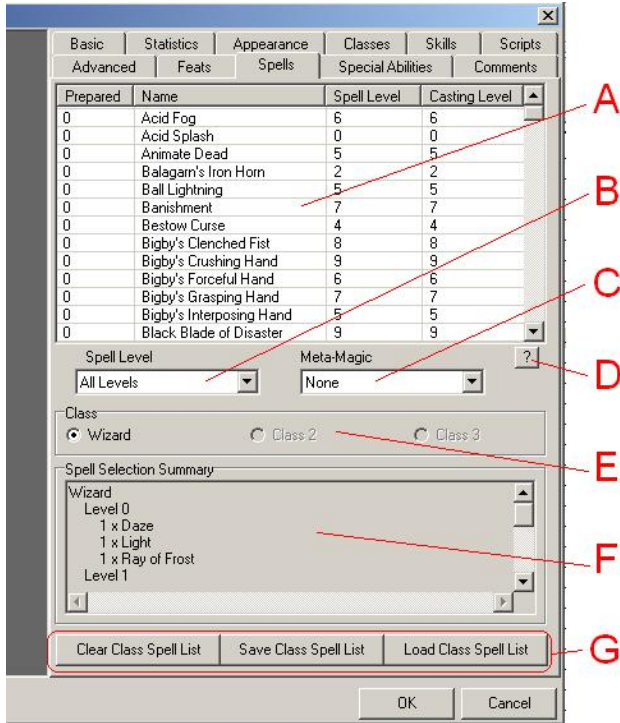
Note:

If you assign a feat to a creature and it does not meet the requirements for that feat it may not use the feat as expected. We strongly advise that you make sure your creature meets all the requirements for any feats that you want it to use.

4.3.9 The Spells Properties of a Creature

In this window you can determine which spells will be assigned to your creature. If your creature is not a spell caster but you still want it to be able to cast spells, then you should modify the special abilities of the creature. (See sub-section 4.3.10.)

Image 4.3.9a – The Creature Properties Screen – The Spells Tab



- A** Here is the spells window. Only the “Prepared” column can be modified so that you can determine which spells are memorized by the creature.
- B** Since the spell list is very long, you can add a filter to see only the spells that are available for a specific level.
- C** If the creature has Meta-Magic feats (like for instance Quicken Spell or Empower Spell) you can use this filter to assign specific spells that will be used in connection with these Meta-Magic feats.
- D** By clicking on this “?” button you will open a small window in which you will have the description of the specific spell or spell category (when there is more than one level to a spell) that is selected.
- E** If the creature is multiclassed and has access to more than one category of spells, then you can switch from one category to the other here.
- F** In the “Spell Selection Summary” you can see what spells were allocated to this creature. This list is adjustable by adding or removing spells in the spells window (see **A**).
- G** Clicking on the “Clear Class Spell List” will remove all spells allocated to this creature under the class selected in **E**. If you press on this button by mistake, then the only way to recover

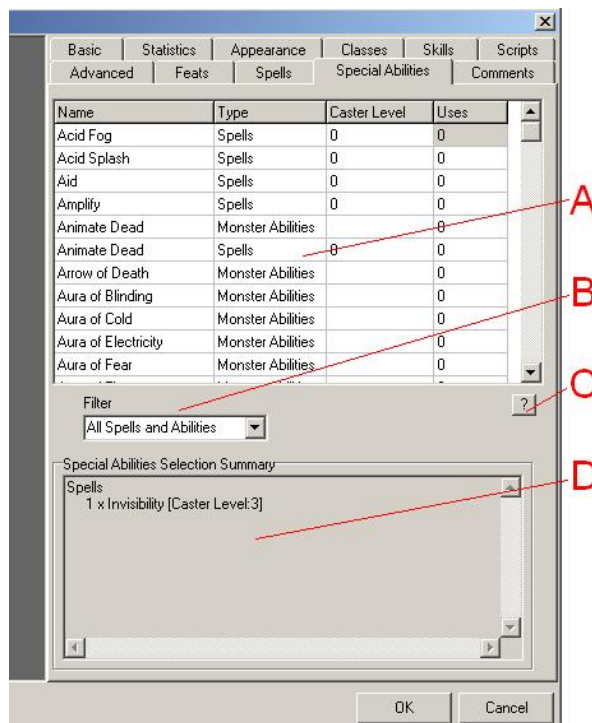
your spells will be by clicking on the “Cancel” button (and even then you will recover only the spells that were allocated to the creature before you opened the Creature Properties Window).

The “Save Class Spell List” button permits you to save the spell list you have made for this creature, under the class selected in **E**, for future use on another creature. The “Load Class Spell List” button will permit you to retrieve a previously saved spell list or one of the default ones created by Bioware.

4.3.10 The Special Abilities Properties of a Creature

In this screen you can add special abilities to your creature. These abilities are in addition to any spells or feats that the creature already has. There is no restriction whatsoever as to the number or type of special abilities you can give to a creature.

Image 4.3.10a – The Creature Properties Screen – The Special Abilities Tab



A This is the Special Ability list. In order to add a special ability to your creature you must add uses in the column specified to that effect.

The “Caster Level” column can also be edited when the ability is a spell. The value to be set in this column must be between 1 and 15 and will be used to determine what effect the selected spell will have. (I.e. if you set the “Caster Level” value to this the effects created by

the spell will be those that would be created if the spell was cast by a spellcaster of that level.)

- B** Since the special ability list is quite long, you can add a filter here if you only want to see either spells or monster abilities in the list in **A**.
- C** By clicking on this “?” button you will open a small window in which you will have the description of the selected spell or monster ability.
- D** In the “Special Abilities Selection Summary” you can see what abilities were allocated to this creature. This list is adjusted by adding or removing abilities in the ability list showing in **A**.

4.3.11 The Comments Properties of a Creature

The large empty box appearing when you click on the “Comment Properties” tab is free for you to use as you wish. You can keep notes in this box that can be useful for future reference for you or any other builder working on this module. Adding content in this box will not have any impact on the creature or on the module.

4.3.12 The Inventory

The Inventory of a creature can be accessed by clicking on the “Inventory” button located at the bottom of the Creature Properties window. You can also access it by right-clicking on a creature placed in an area.

Image 4.3.12a – Inventory Contents – Standard Equipment

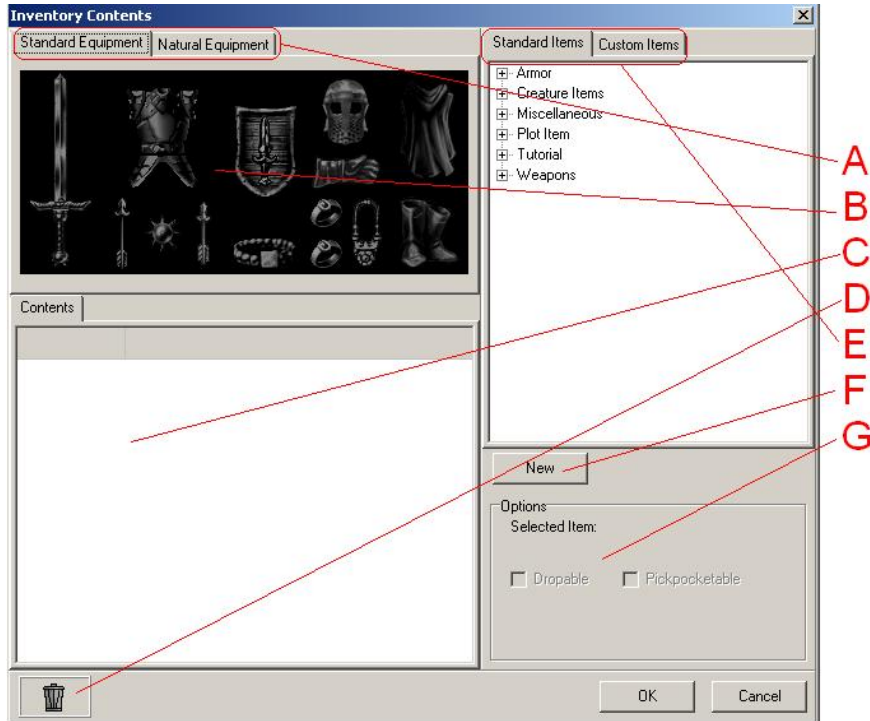
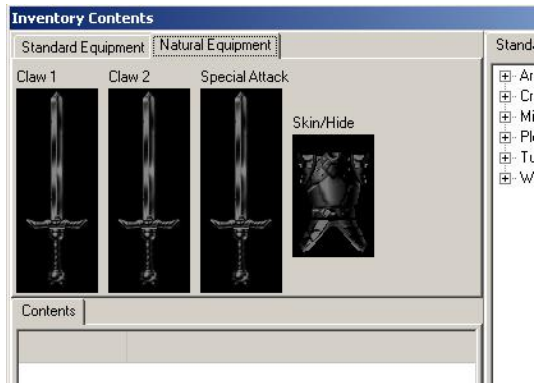


Image 4.3.12b – Inventory Contents – Natural Equipment



- A** Using these two tabs you can select what type of item you want to equip on your character. Standard equipment is the usual equipment screen as seen when playing NWN (swords, shields, etc.) while natural equipment is the screen where you can add creature items (skins, fangs, etc.).

You can add either or both standard and natural equipment to any creature, even if the appearance of the creature does not provide for the visual ability to wield such equipment (i.e. a wolf can wield a greataxe and a gnome can have claws).

If a creature wears both standard armor and a skin/hide, the effects of both items will stack. Example: if a skeleton wears padded armor (armor class 1) and a skeleton hide with an AC+1 item property, then the skeleton will have of bonus of 2 to its armor class.

If a creature wields both a standard weapon and natural attack items, the standard weapon will be used and the effects of the natural items will not stack. The only exception to this seems strangely to occur during attacks of opportunity, in which case the properties of the standard weapon and of one of the natural attack items (chosen randomly) seem to stack.

If natural attack items are assigned to more than one item slot (i.e. Claw 1, Claw 2 and Special Attack), the properties of only one of these items will be considered on a successful hit, chosen seemingly randomly by the engine (the Special Attack item being chosen less frequently than the Claw 1 and Claw 2 item). If only one natural attack item is assigned to a creature, the properties of this item will be applied to every successful hit made by the creature.

- B** These are the standard item slots for your creature.
- C** You can add any item you want your creature to hold in its inventory in this section. The inventory of a creature is usually used to stack potions, scrolls and ammunitions, but it may also be used to carry any treasure you want your creature to drop when it gets killed.
- D** You can drag and drop an item to this trash can if you want to remove it from the creature's inventory.
- E** These are the item palettes from which you can select items for your creature. The "Standard Items" palette contains all standard equipment and also magical items created by Bioware. The "Custom Items" palette contains all the items you have created and, if CEP is installed on your module, all CEP items.
- F** Clicking on the "New" button will launch the Item Wizard (see Section 7.1).
- G** These options apply to the item you have selected in the creature's inventory. If the "Dropable" box is enabled, then the creature will drop this item when it will be killed. If the "Pickpocketable" box is enabled, then the any player performing a successful pickpocket attempt on the creature can take the item.

4.4 The Factions

The faction property of a creature will determine how it will react to player characters or to other creatures. This is one of the properties of creatures that can produce many unwanted effects, and it is therefore very important to consider overall effects that can be caused before adding new factions or changing the properties of standard factions on your module.

There are four standard factions that come with every module:

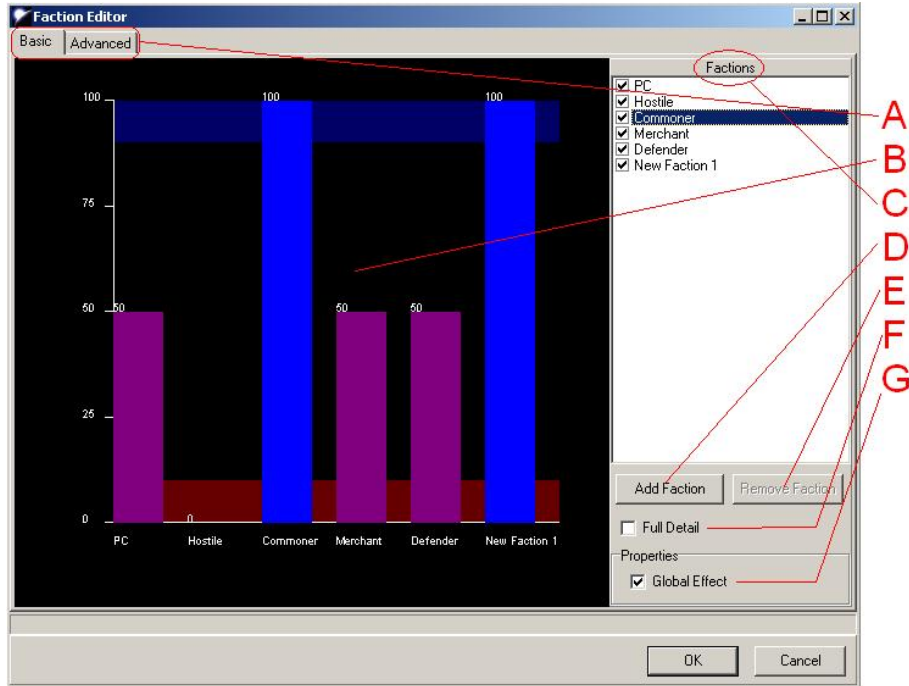
- 1) Hostile: This faction is used for all creatures that are intended to be enemies of **all** other factions (except of course other hostile creatures – they will not attack one another).
- 2) Commoner: This faction is used for all creatures that are intended to behave in a “cowardly” fashion.
- 3) Merchant: This faction is used for creatures that are basically neutral to the other factions, except towards hostile creatures.
- 4) Defender: This faction is used by Bioware for creatures that are intended to help creatures of other factions in need of help (like city guards or soldiers).

Every faction has settings that establish how it reacts to other factions (i.e. whether a faction is friendly, neutral or hostile vs. each other existing faction). These settings are called reputation points.

The number of reputation points of a faction is customizable (see below for more details about how to change the settings of a faction). It is however very, very important before you modify anything about the standard factions to consider the fact that every pre-made creature that you will find in the standard palette is of one of these 4 standard factions. If you modify the settings of the Hostile faction, it will impact all standard hostile creatures spawned through any encounter.

TIP: When you have opened the Faction Editor but have not made any modification, it is always preferable to close the editor by clicking on the “Cancel” button. That way, you will avoid the toolset recalculating the factions for no reason.

Image 4.4a – The Faction Editor – Basic Tab



A Here you can select the “Basic” or “Advanced” way of showing the settings of each faction (see Image 4.4b for the Advanced tab screen). The information appearing under both tabs is the same, it is simply presented in different ways.

B This graphic represents the number of reputation points that the selected faction in “**C**” has towards every other faction **at the beginning of the module**. The actual number of points can be changed in game by the actions of members of these factions and through scripts.

To modify these settings, simply drag the column to the desired number of reputation points. There are three levels of setting:

- i) 0-10: a setting between 0 and 10 reputational points confirms that the faction selected in “**C**” is hostile towards the faction named under the column. The faction selected in “**C**” will therefore attack any member of the other faction on sight.
- ii) 11-89: a setting between 11 and 89 reputational points confirms that the faction selected in “**C**” is neutral towards the faction named under the column. The faction selected in “**C**” will therefore neither attack or help the members of that other faction.
- iii) 90-100: a setting between 90 and 100 reputational points confirms that the faction selected in “**C**” is allied with the faction named under the column. The faction selected in “**C**” will therefore help the members of that other faction that are under attack.

The actual number you choose for these settings is important since it will determine when the faction switches from ally, to neutral to hostile.

Example: You add a script on a chest owned by a NPC that provides that if a player opens it, the player loses 10 reputation points towards the faction of the NPC. If the setting of the faction of this NPC towards the PC was set in the screen above at 15, then the NPC will turn from neutral to hostile towards player characters if the player opens the chest. If however the reputation level was set to 50, then after this script has run the reputation level will now be at 40 and the NPC will still be neutral.

It is important to note that reputation levels are not necessarily reciprocal. Faction A may be friendly towards Faction B and Faction B may act neutral towards Faction A. It is therefore very important when you create a new faction to establish first how this faction reacts to the other ones, but then also how the other factions react to your newly made faction.

- C** Here you can determine which faction you wish to view the reputation levels of by clicking on it. You will then have in the screen identified in “**B**” how this selected faction reacts to every faction that is ticked. You might want to untick some factions when you want to have a clearer view of the reputation levels.
- D** Clicking on this button will permit you to create a new faction. The toolset will then ask you which one of the standard factions you want to use as a template for your new faction. No matter which one you will use, you will then have the opportunity to customize the new faction. Note that the choice of template faction can have an effect if the creature is exported and the “Reset Factions to Parent Factions” box is ticked.
- E** Clicking on this button gives you the option of removing a faction that you no longer use. If some creatures are still members of a faction that was removed, they will then switch to the template faction you used when you created the custom one.
- F** By enabling the “Full Details” option, a green column will appear next to each one of the faction columns in the faction editor. This green column gives you the number of reputation points the faction named under the column has versus the faction selected in “**C**”. This permits you to have a full picture of how the selected faction reacts to other factions, and how these other factions react to the selected one as well.
- G** By enabling the “Global Effect” property, any action taken by a player (or any other creature) against one member of the selected faction will affect how all members of this faction feels about the said player. If the “Global Effect” property is not ticked, then only the members of the affected faction that are in perception range will react to the action of the player.

Example: A player is alone in a house with a NPC member of the faction Z. The player decides for some reason to attack this NPC. The NPC therefore turns hostile towards the player. After the fight, the player leaves the house. If the “Global Effect” property was enabled for the faction Z, then all members of this faction will now on be hostile towards the player. If the “Global Effect” property for this faction was disabled, the other members of this faction will act as if nothing had ever happened.

This property is very important as it determines how much freedom you want to leave to the players in your module. In the example mentioned above, both possibilities have their strengths and weaknesses. Enabling the “Global Effect” property will force players to think

twice before taking any action against a faction. The downside however may be on the logical side: if the player is alone with this NPC, how can the other members of the faction know what the player has done and therefore react to his actions? On the other hand, disabling the “Global Effect” property will serve logic. Only the NPCs that are a witness of the player’s actions will react to these actions. The downside may however be that players acting evil will never get caught and have to respond for their actions.

As mentioned above, the “Advanced” tab simply shows you the information appearing in the “Basic” tab screen in a different way. By clicking on this tab, you will get the following screen:

Image 4.4b – The Faction Editor – Advanced Tab

The screenshot shows the 'Faction Editor' window with the 'Advanced' tab selected. It features a table with reputation points between factions and a list of selected factions.

	PC	Hostile	Commoner	Merchant	Defender	New Faction
Hostile	0	100	0	0	0	0
Commoner	50	0	100	50	50	100
Merchant	50	0	50	100	100	50
Defender	50	0	100	100	100	100
New Faction	50	0	100	50	50	100

Below the table, a list of factions is shown: PC, Hostile, Commoner, Merchant, Defender, and New Faction 1. Red callout boxes labeled H, I, J, and K point to the 'Hostile' column, the 'Hostile' row, the 'Commoner' column, and the 'Factions' list respectively.

- H** This column of the table is the list of factions that are customized when you change the number of reputation points shown in the table. In other words, this is the “how does the faction react” column. If you look at the first faction in this column, the Hostile faction, you will see that it has 0 reputation points toward the player characters (and is therefore hostile towards players), 100 points towards the Hostile faction (and therefore is allied to the other members of this faction), 0 points towards the Commoner faction (and is therefore hostile to members of the commoner faction), etc.
- I** This column of the table is the reference list of factions. See the explanation provided in “H” on how this reference list is used.
- J** The numbers appearing in the table are the number of reputation points given to the faction listed in “H” towards the faction in “I”.
- K** The factions that are selected here are those that will be used to generate the table. To select a faction, simply left click on the faction once. To select a range of factions, left click on the first faction of the range, hold the Shift key on your keyboard and click on the last faction in the range that you wish to select. You can also select a number of factions by left-clicking on them while holding the CTRL key on your keyboard.

Dallo Officer 02 July 2005

*CKR had a *lot* of faction problems early on, and whilst I managed to resolve most of them, there are still issues with factions that are unclear in my head and I wonder whether some of you gurus can enlighten me.*

a) Hench assumes PC faction once they've signed up. So sayeth the literature.

**beeeep* Wrong.*

Hench in CKR were originally standard defender and were sometimes attacked by commoners once henchified. Seemed to happen, when it did, just after medium to large battles. Possible explanations? (Hench spells did not harm neutral/friendlies). For awhile these commoners would not attack the PC but after awhile they seemed to figure that the PC and hench were connected and would then attack the PC also. So it was a double glitch...

b) Neutral (ie 50% on all other factions) factions are, literally, neutral to everyone.

**beeeep* Also wrong.*

It almost seemed as though it depended on who was perceived to be attacking first. So a non-global totally neutral character would often attack either PC or hostiles depending on the situation. Explanation??

C) Question: does it matter which faction is the parent faction when making a custom one? Is there a ghost in the machine even though all sliders have been manipulated??

Will a "neutral" NPC react differently because of its faction's parent faction even though everything is set at 50%??

Mermut Administrative Team Moderator Saturday, 02 July 2005

I've found that the parent faction DOES matter. I've had the best luck (ie least amount of 'inherited' behaviour) when starting with commoner or merchant as the parent faction. One of my first experiences with this was when I made a 'wild animal' faction which was neutral to all. I'd originally had the parent faction be hostile. The animals wouldn't auto attack the PCs or the guards... but the guards (defender faction) DID try to exterminate the badgers, despite an allegedly neutral reaction to them. When I redid the faction (same exact relations) using the merchant faction as the parent, everything behaved as I told it to/expected it to.

4.5 Designing Tips

Creatures are highly customizable. A lot of freedom is given to builders in this area. Here is a list of tips you might want to consider when you will be customizing your creatures.

While creating and editing creatures, try to remain faithful to what the creatures actually are: In the toolset you can basically give the appearance of a dragon to a very weak creature. You must however be aware that most players will not like that since it is simply not logical. (A 30-foot tall dragon should not be killed after two or three sling shots!) If your intention is to build a low level module, it is preferable to use weak creature appearances for your custom creatures. If your intention is to build a module involving trolls, dragons and demons, than you might want to consider building a module for mid or high level player characters instead of low level ones.

Remove low-level spells on high-lever spell casters: If you do not want a creature to cast low-level spells that were assigned to it upon creation because they might have better melee combat abilities

than a "Ray of Frost" spell, then simply delete them. A spell caster creature does not need to have a full book of spells.

4.6 How to...

4.6.1 ...add animations to my NPCs to make them more alive

There are many ways to do this. The easiest one is the following:

- 1) Make sure that your creature still has one of the two default OnSpawn scripts under the "Scripts" tab of its properties. These two default scripts are " x2_def_spawn" and " nw_c2_default9". If you have not touched the scripts of your creature, then you can be sure that its OnSpawn script is one of these two.
- 2) Add an integer variable (i.e. int) on the creature by clicking on the "Variables" button under the "Advanced" tab of the creature properties. The name of this integer variable should be X2_L_SPAWN_USE_AMBIENT if you want your creature to be able to move around in the area or X2_L_SPAWN_USE_AMBIENT_IMMOBILE if you want it to remain immobile. Then give this integer variable a value of 1.

If you want to get into more detailed NPC behaviour and if you have some basic knowledge about scripting, then you can also:

- 1) Make sure that your creature has the " nw_c2_default9" default OnSpawn script.
- 2) Click on the "Edit" button next to this script to open the script editor. You will then notice that this script is heavily commented. Read these comments and uncomment (i.e. remove the "//" signs at the beginning of the line) the commands you want to apply to the animation behaviours of your creature.
- 3) Save your script, but make sure to give it another name by clicking on the "Save As" button or otherwise the behaviour you are creating will apply to all creatures using the default script.

If you are looking for animations for your NPCs that are even more detailed, then we suggest you to use the scripts made by Deva Bryson Winblood that are very well documented, which can be found here:

<http://nwwvault.ign.com/Files/scripts/data/1066589562597.shtml>

or this very detailed tutorial written by Ayath The Loafer and Nereng relating to the ambient animations written by Bioware for the HotU expansion but that never made it to the game:

<http://nwwvault.ign.com/Files/scripts/data/1083921343000.shtml>

4.6.2 ...make a dead creature.

The default game script for this is "x2_spn_dead" place this script on the creatures OnSpawn event and clear all scripts from its other events and you will have a nice decorative corpse.

4.6.3 ...make my creatures drop crafting items.

Make sure your creature has the OnDeath event script set to NW_C2_DEFAULT7 or if you are using a custom OnDeath script call `craft_drop_items(oKiller)`; see the example below.

```
#include "x2_inc_compon"
#include "x0_i0_spawncond"
void main()
{
    object oKiller = GetLastKiller();
    craft_drop_items(oKiller);
}
```

CHAPTER 5 Conversations

Writing conversations is a very challenging aspect of building a module. It will mostly be in conversations that your story will take form. There are no templates, wizards nor generator tools that will help you to write good conversations. You are the sole storyteller and you decide how your conversations will shape your module.

In this chapter we will try to explain each aspect and function of the Conversation Tool so that you can write conversations in the toolset in the most efficient way. Although we cannot provide you the words with which to write your conversations, we will provide you with the details on what the Conversation Tool has to offer.

5.1 *Creating a Conversation File*

5.1.1 Writing your conversation

There are many different ways to create your conversations and no single way is really better than another. Each has its own benefits. The best way to create a conversation is by using the means that you prefer. Here are some examples of what to use when creating conversations:

- i) Pen and Paper;
- ii) Word Processing Program;
- iii) Toolset Conversation Editor.

These are just a few of the many viable mediums you can use. It is preferable to try many different options before deciding which one works best for you.

A word of warning: Writing your conversation directly in the Toolset Conversation Editor without first knowing what you will write can be VERY counter-intuitive.

5.1.2 Integrating your conversation in the toolset

In the toolset, conversations take the form of a Conversation file. These files are created with the Conversation Editor, which can be accessed by:

- i) Selecting the “Conversation Editor” option in the “Tools” menu of the toolset;
- ii) Clicking on the “edit” button next to the conversation property of a creature, of a placeable object or of a door; or
- iii) Right-clicking on a creature and choosing the “Conversation” option.

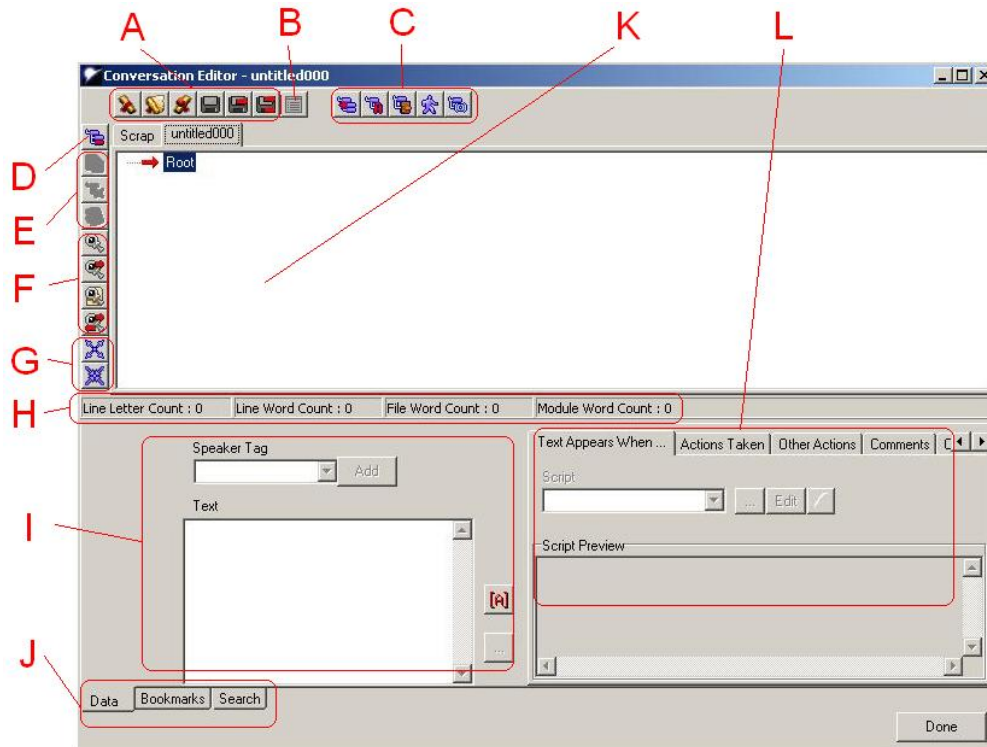
For the options ii) and iii) above, if a conversation file has already been assigned to the selected object, that conversation will be opened in the Editor upon launching. If however no conversation file was assigned to the selected object, then the Conversation Editor will be launched and any conversation will you write and save will automatically be assigned to the selected object.

Note: you can use the shortcut keys CTRL+C for copy and CTRL+V for paste.

5.2 The Conversation Editor

Unlike many other tools in which the properties are organized under different tabs and categories, most properties and functions of the Conversation Editor are found on its opening screen. The following image shows you how these properties and functions are grouped.

Image 5.2a – The Conversation Editor Screen



- A** These buttons are the standard file commands. From left to right, they provide the following commands: "Create a new file", "Open an existing file", "Close the current file", "Save the current file", "Save as" and "Save All".
- B** Clicking on this button brings you to the options screen of the Conversation Editor. The options provided in this screen will be fully covered in Section 5.5.
- C** Enabling one or more of these buttons will highlights nodes in the conversation that contain, from left to right, comments, actions, quests, animations and sounds, as more fully explained in Section 5.5 below.

- D** This button is used to add a conversation node to your conversation, which will be covered in Section 5.3 below.
- E** These buttons are the standard edit commands (i.e. the “copy”, “paste” and “cut” commands).
- F** These buttons are the standard find commands (i.e. “Find text”, “Find next”, “Find in files” and “Replace”). The find text button will search only in the current conversation file. If you want to find text in all open files or in all conversation files of your module, you need to use the “Find in files” command.

When you use the “Find in files” command, the results of the search will be accessible by clicking on the “Search” window tab identified by the letter **J** in Image 5.2a. The results will remain in this window until a new search is performed. To access the found nodes in which the searched text appears, double-click on any search result line in this window.
- G** These buttons are the “Expand All” and “Collapse All” buttons. Clicking on these buttons will either expand all the nodes in your conversation tree or minimize them.
- H** For the statistics fan in you, here you can find out how many letters or words there are in the selected line, file or module.
- I** In this window you can edit the text of any node contained in the conversation file. The options provided in this screen will be fully covered in Section 5.3.
- J** These tabs permit you to access the “Data” properties (the ones appearing in Image 5.2a), the “Bookmarks” you have set (see Section 5.5 for full details) and the “Search” results (see comments made under letter **F** above).
- K** Your conversation tree is in this window. How to create this tree and a description of its structure are topics covered in Section 5.3 below.
- L** This window provides you with the properties of any selected conversation node and the properties of the conversation files, which are more fully explained in Section 5.4 below.

5.3 The Structure of a Conversation

5.3.1 Creating your conversation tree

Conversations in the Editor are constructed in the form of a tree, starting from the root and expanding in unlimited nodes.

The root is basically the starting point of the conversation. It can easily be found since it is always at the top left corner of the conversation (see Image 5.3.1a below).

The nodes are each line of text that you have attached either to the root of the conversation or to another node. When the text is spoken by the “Owner” of the conversation (i.e. the creature, placeable object or door to which the conversation file is attached), the conversation node will appear in red in the Editor. When the text line is to be spoken by the player, it will appear in blue.

The following image shows you an example of the first nodes of a conversation file being written:

Image 5.3.1a – Conversation Editor Screen – The “Add” button



As mentioned above, the node in red is the sentence that will be spoken by the NPC when a player clicks on it. The player will then have 3 choices for a response, which are the three nodes in blue.

The word “OWNER” at the beginning of the red node means that this sentence will be spoken by the object to which the conversation file has been assigned. As we will see in sub-section 5.3.4, you can change this property so that another object speaks a specific node and thereby create a multi-party conversation. By default, each red node will be assigned to the owner of the conversation.

The words “END DIALOGUE” at the end of each blue node means that the dialogue tree stops at this point in the conversation. While we want the conversation to stop at the “Nothing. Thanks.” answer, for the other two we probably do not want the conversation to end there. In order to add a new node that will be spoken for each one of these two questions, you have to select the blue node where you wish to add a new node in the conversation tree and then click on the "Add" button as shown in Image 5.3.1a. An “Input Text” window will then pop up in which you can type the text for the new node.

The rest of the process of creating a conversation is simply to add new nodes to your tree until all options and possible answers in your conversation are covered.

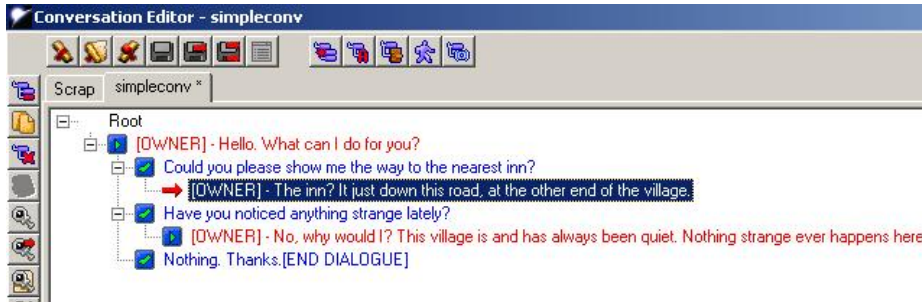
5.3.2 Using links

Unless you want your conversation tree to become gigantic, it will be necessary for you to use links to avoid any unnecessary recurrences in your conversation file.

Links allow you to send a player from one branch of your conversation tree to another one.

In the example showed in Image 5.3.2a below, after the NPC has provided an answer to the first question of the player, you might want to give the opportunity to the player to ask the second question that was listed in his answer options at the beginning of the conversation tree. Instead of typing the “Have you noticed anything strange lately” question again as a new node under the provided answer to the first question, you can add a link to the previous node.

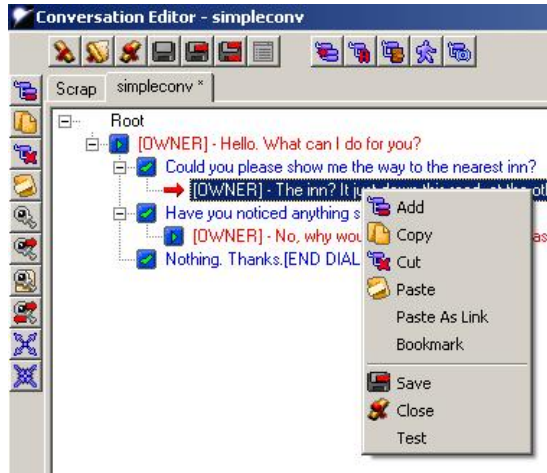
Image 5.3.2a – Conversation Editor – Links Part I



In order to add this link, you must:

- a) Select the “destination” node (i.e. the node you want your link to refer to);
- b) Click on the “Copy” button on the left toolbar, or type CTRL-C;
- c) Select the “origin” node (i.e. the node where you want to add your link); and
- d) Right-click on the origin node and select the “Paste As Link” option in the dropdown menu as show in Image 5.3.2b.

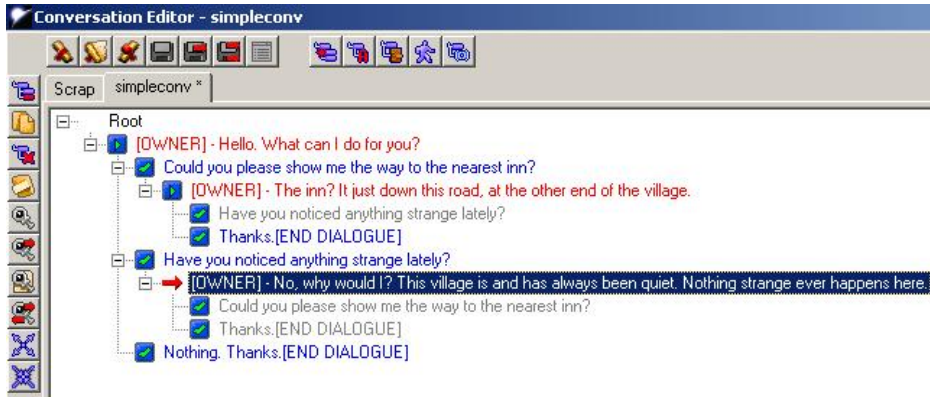
Image 5.3.2b – Conversation Editor – Links Part II



You will then see the destination node appearing in grey under the selected node. This confirms that the link to the destination node has been added to your conversation tree.

The conversation as shown in Image 5.3.2c below shows you the final loop that can be done in this simple conversation to ensure that players have the opportunity to ask the two possible questions to the NPC without the need to type both questions twice.

Image 5.3.2c – Conversation Editor – Links Part III



Note that links can also refer to a node that is spoken by the owner of the conversation. As an example, if you want a NPC to answer in the same fashion to two different questions asked by a player (or to the same question but asked in a different way later on in the conversation tree), then you can link the red node by using the same method as the one described above.

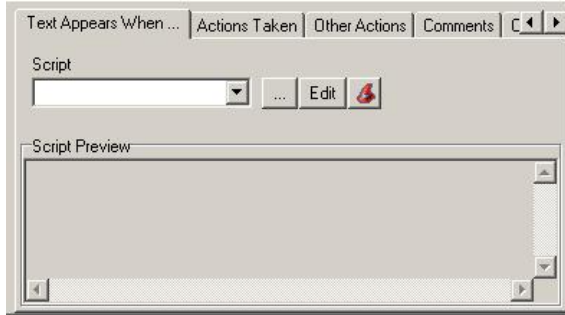
TIP: If while reviewing your conversation file you want to locate the “destination” node of a node that was added as a link rapidly, double-click on the grey node and you will be brought to the destination node automatically.

5.3.3 Adding conditional nodes

You might want a NPC to say a specific line if the player is a male and another thing if the player character is a female. You might also want some specific nodes to be available to the player only if a specific quest has previously been completed. These are only two examples of situation where you would want conditional nodes to be added to your conversation.

In order to create a condition node, you must first type it as any other standard node in your conversation tree. Once you have finished writing your node and while it is still selected, you can add the condition under which you want this node to appear in the “Text Appears When...” property at the bottom right corner of the Conversation Editor as shown in Image 5.3.3a below.

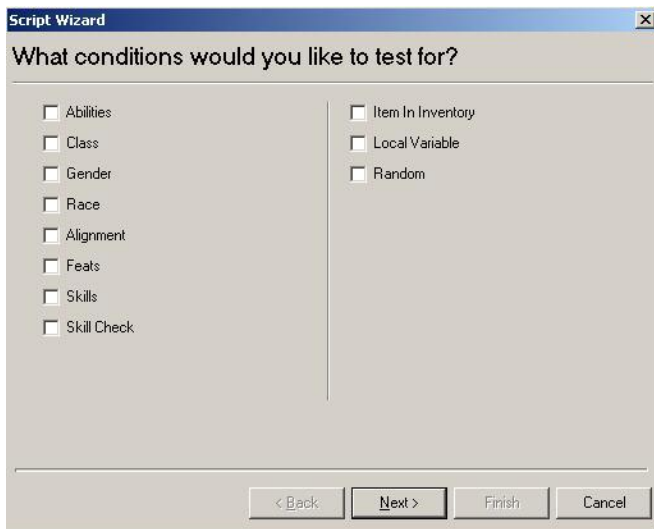
Image 5.3.3a – Conversation Editor – “Text Appears When...” Property



As you can see, adding such a condition must be done through a script. Clicking on the “Edit” button will launch the Script Editor for you to write this script.

If you do not know how to script or if you have limited knowledge in that area, you might want to use the Script Wizard by clicking on the icon showing a red wizard hat. If you do, the following screen will appear:

Image 5.3.3b – Script Wizard



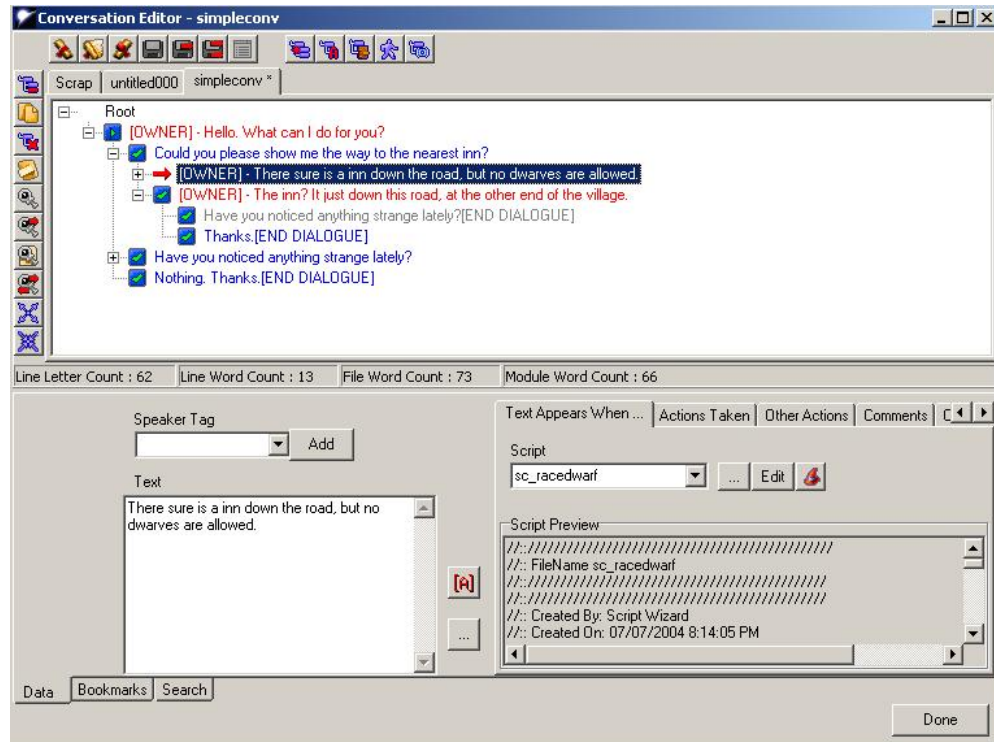
As you can see, using the wizard permits you to pick from many criteria categories to use as conditions for the node in the conversation. You can choose as many condition categories you wish in this screen.

Once you have selected the conditions that will be tested and have pressed the “Next >” button, you will see further screens that will provide you with more options (examples: the selection of races that will be checked, skill checks that can be run, etc.) to define exactly the conditions that will be provided. Once you have defined these conditions, you will be asked to choose a name for your

script. There is a general naming convention that provides that such scripts start with “sc_”, for “starting conditional”. The rest of the name is for you to provide. We recommend that you complete the name with simple data that will help you to remember the conditions that are checked simply by looking at the name of the script. As an example, “sc_racedwarf” would be the script that makes the node appear under the condition that the player character is a dwarf.

Once you have added your script, the conversation could look something like this:

Image 5.3.3c – Conversation Editor – Conditional Node



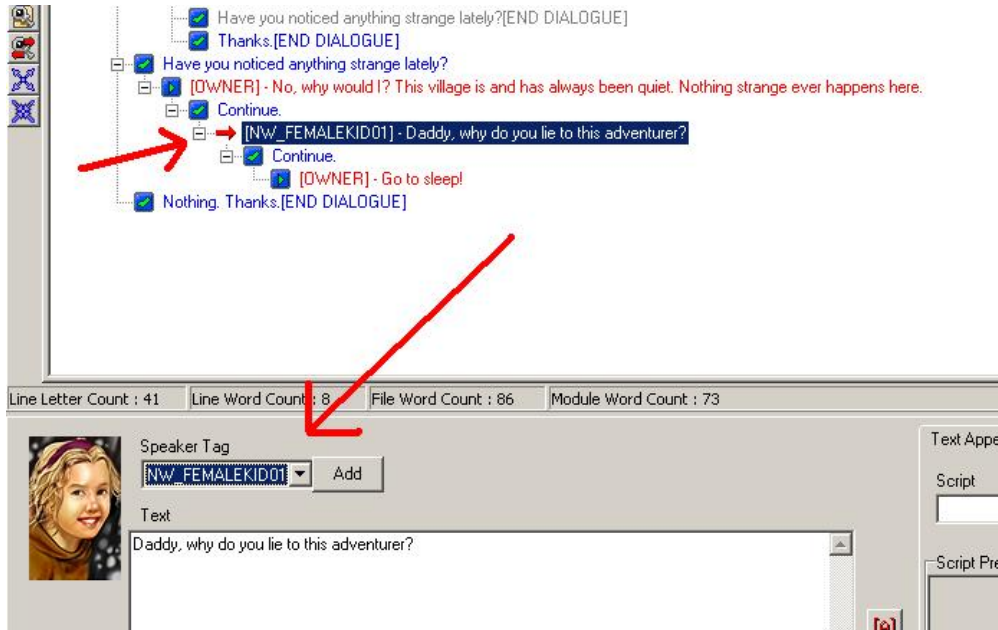
As you can see in this conversation, when the player asks if there is an inn nearby, there are two possible answers for the NPC: one for players that play dwarves and one default answer for any other player.

WARNING: The engine always reads nodes from top to bottom. Therefore, if you want your conditional nodes to appear when the condition is met, it is important for this conditional node to appear before the default one. To move your nodes to respect this rule, you can drag and drop any conversation node to any position in your conversation tree.

5.3.4 Creating a multi-party conversation

As mentioned above, the word “OWNER” at the beginning of any red node confirms that it will be spoken by the owner of the conversation (i.e. the NPC to which it is attached). You can however change the speaker of a specific node to include a third party into a conversation.

Image 5.3.4a – Conversation Editor – Multi-Party conversation



In order to do that, you select the node that you want to be spoken by a third party and change the “Speaker Tag” as shown in Image 5.3.4a. The dropdown list under the “Speaker Tag” will provide you with the tags of all objects in the area. Be careful however: although the portrait of the third party speaker will appear in your conversation tree, if the third party speaker is too far away from the owner of the conversation, the player involved in the conversation will see a generic portrait and the nodes will appear to him to be spoken by an “Unknown Speaker”. It is therefore necessary for the third party speaker to be close to the owner of the conversation for this to work properly.

If the third party speaker is not placed in the area where the owner of the conversation is, you can click on the "Add" button to add the tag of any blueprint creature, door or placeable. This allows for the possibility that the third party speaker might be created later on through a script.

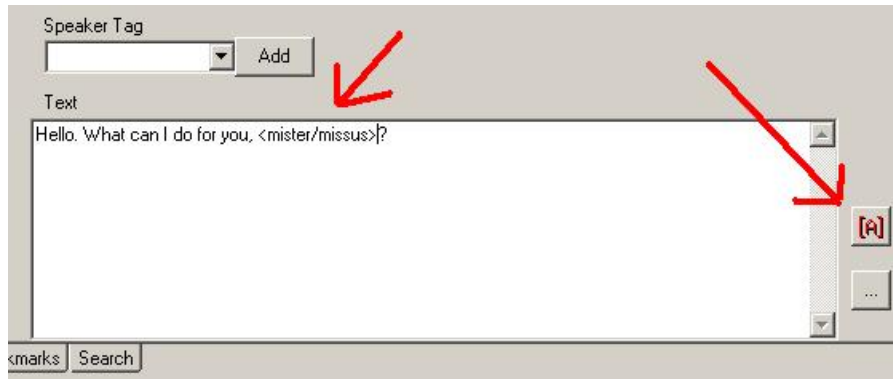
Be careful when you write the rest of the conversation tree after you make a third party speaker participate. The next red node that you will create in this branch will automatically be assigned to this third party speaker. If you want the owner of the conversation to speak again in this conversation branch, then you must change the “Speaker Tag” back to the blank tag at the top of the dropdown menu.

5.3.5 Inserting tokens and highlighting text

a) Standard Tokens

Tokens permit you to include some text that will adjust itself automatically depending on the characteristics of the player involved in the conversation, without the need to create conditional nodes to cover all possible characteristics. A few examples of situations where you might consider using a token are when you want your NPC to say Mr. to a male player character and Mrs. to a female one, or if you want the NPC to refer to the class or race of the player character involved in the conversation.

Image 5.3.5a – Conversation Editor – Tokens



To access the list of tokens click on the red “[A]” button as shown in Image 5.3.5a above. You will see your token appear in the Text window where your cursor was when you opened the token list.

Another way to access the list of tokens while you are writing a node is by hitting the menu key on your keyboard (it may be in different places on different keyboards, but on a Compaq keyboard it is the third key to the right of the space bar).

Do not forget to add spaces before and after the token if it is inserted in plain text, the Conversation Editor does not do that automatically.

b) Scripting custom tokens

You can also create your own tokens through scripting. This can be useful when you want an NPC to speak a certain number that is stored as a variable or when you need to use more complicated tokens.

In order to do this, you must first create the token in a script that is run before the node that will refer to this token is spoken. The script can be run in a previous node in the tree, in the OnConversation script of the owner of the conversation, or anywhere else as long as it is before the token is called in the conversation.

To show you how to script a custom token, we will use a simple example. In a module, the player is asked by a mayor to go to a forest and cut some trees. You track the number of trees that are

bashed by the player by placing a script incrementing a local variable set on the module called "cuttree" in the OnDeath event of the tree placeables. Then you add the following script in the "Action Taken" event of the first node in the conversation of the mayor:

Script 5.3.5a – Creating a custom token

```
void main()
{
int nTree = GetLocalInt(GetModule(), "cuttree");
string sTree = IntToString(nTree);
if(nTree==0) SetCustomToken(200, "no trees");
else if(nTree==1) SetCustomToken(200, "one tree");
else SetCustomToken(200, sTree+" trees");
}
```

The first variable in the SetCustomToken() command is the number of the custom token. This can be any number. You might however want to track the number you use in a module in order to avoid duplicate custom token numbers.

After you have created the custom token, you can call it in a conversation at any time by adding it in the following format: <CUSTOMXXX>, where XXX is the token number you have given to your custom token in the script that created it.

Therefore, in the previous example, let's say that the mayor greets the player with the following question: "So, brave adventurer, how many trees have you cut for me?". In the answer node you could then add the following: "I have cut <CUSTOM200>." The token will then be replaced by "no trees", "one tree" or "X trees", depending on the variable that is tracking the number of trees that are cut.

c) Highlighting text

You will notice when you access the list of tokens that, at the very end of the list, there are the following tokens: "StartAction", "StartCheck", "StartHighlight" and "/Start". These are not tokens, actions nor checks, but rather ways to change the color of the text in a node. The "StartAction" token will change the text that is written after the token to green, the "StartCheck" will change it to red and the "StartHighlight" token will change it to blue. The "/Start" token must be used to turn back the text to white color. Note that the blue text is nearly unreadable in most circumstances.

Another way to add these false tokens into a node is, once you have opened the token list, to click on the "Highlight" tab at the top of the token window. You can then type the words that you want to appear in another color.

The reason why these false tokens used the words "Actions", "Check" or "Highlight" is because these are the code colors used by Bioware in the original campaigns. You might remember while playing these campaigns that every skill check in conversations was written in red. Of course, you do not have to follow this color convention and you can use these color tokens as you see fit.

5.3.6 Creating conversations in multiple languages

You will notice a “...” button at the bottom right corner of the text window of the Conversation Editor. (See Image 5.3.5a for reference.) A similar button can be found everywhere there is text to be input (i.e. creature name, module description, etc.). By clicking on this button, you will open the “Edit String” window.

In this window, you can add text that will be seen by any player that runs a foreign version of NWN. You will notice that, by default, any text typed in the toolset will be assigned to the English version of NWN and the text for the other versions will be left blank. This does not need to be modified as long as your module will be available in only one language. If however it is your intention to translate your module in other languages, you will have to assign written text to the appropriate language window.

Switching from the Conversation Editor general screen to the Edit String window for every node that you will be writing will rapidly become very tedious. In order to avoid this, it is recommended that you first write your module in the language you prefer and then use the NwnMultiLang tool created by gTach to export all the text included in your module to a .txt file for translation:

<http://nwwvault.ign.com/Files/other/data/1064008824326.shtml>

Note that for a player to see your translated text in a foreign language, the player must own the foreign version of NWN.

Note: you can use the shortcut keys CTRL+C for copy and CTRL+V for paste.

5.4 The Properties of a Conversation Node

At each node of a conversation you can have actions performed, whether it is playing an animation, adding journal entries, having the speaker turning hostile or simply adding a local variable to keep track of the evolution of some quest. The following sections will provide you with the details on how and where such actions can be added to your conversation file.

5.4.1 The “Text Appears When...” Property

The “Text Appears When...” property of a conversation node permits you to make a node conditional. This topic has already been elaborated upon in Sub-section 5.3.3 of this manual.

5.4.2 The “Action Taken” Property

The “Action Taken” property permits you to run a specific script when the selected node is being spoken in the conversation.

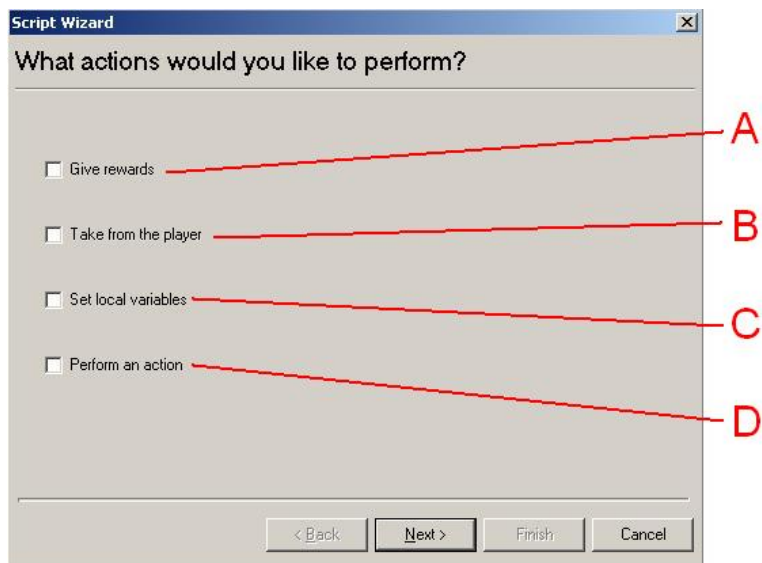
Be careful: the script that you add in this property will only run if the node to which it is attached is spoken. Therefore, if you add a script on a blue node (i.e. a node that can be spoken by the player), the script will only run if the player chooses to speak this node.

Image 5.4.2a – Conversation Editor – Action Taken Property



You can attach an existing script to a node by clicking on the “...” button. If you click on the “Edit” button, then it will launch the Script Editor for you to write the script. If you click on the red wizard hat icon button, then you will launch the Script Wizard that will write the script that will be run for you.

Image 5.4.2b – Script Wizard – Action Taken Property



As you can see, unlike the “Text Appears When” script wizard, the wizard attached to the “Action Taken” property does not provide you with many choices for scripts.

- A** The “Give rewards” action will permit you to give gold, experience points or items to the player.
- B** The “Take from the player” action provides the opportunity to take gold, experience points or items from the player.

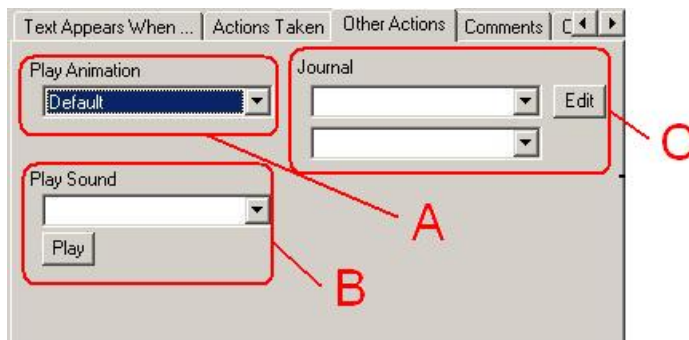
- C** The “Set local variables” can be used to store variables on the player to keep track of the evolution of a quest, for example.
- D** The “Perform an action” option allows you to ask the speaker to attack, to launch a merchant (and there are easier ways to create a merchant as will be seen in Chapter 14) or to modify how the faction of the speaker feels about the player.

Fortunately for those who have very little or no scripting knowledge, these are the most common actions that need to be performed in a conversation. But if you want a more complex set of actions to be performed, then you will have no other choice but to get dirty and write the script by yourself or to use one of the script writing applets available from [Neverwinter Vault: Lilac Soul's Script Generator](#) or [ScriptEase](#) are both good applications.

5.4.3 The “Other Actions” Property

The “Other Actions” property of a node is one of the most interesting things to play with if you want to add details to your module, and increase the overall visual and audible experience of the players that will be playing your module.

Image 5.4.3a – The Conversation Editor – The Other Actions Property



- A** Play Animation: The “Play Animation” dropdown menu provides you with a list of visual animations that can be performed by the speaker of the node. Some of them (like “Read” or “Salute”) will be performed once, at the moment the node is spoken (they are called “Fire Forget” animations) while some others (like “Talk Laughing” or “Bow”) will be performed by the speaker until another node is spoken (they are called “Looping” animations). All these animations are strictly visual: they do not affect the game otherwise. Just as an example, if you add a “Drink” animation to a node that is spoken by the player, it does not mean that the engine will remove a potion from the player’s inventory.
- B** Play Sound: The “Play Sound” dropdown list provides you with a huge list of sound files that can be emitted by the speaker of the node. The good thing is that you can hear the sound that are you adding on a node by clicking on the “Play” button. The bad thing is that there is no search tool and therefore you will have to scroll the dropdown list, often for quite a long time, before you can find the sound you are looking for.

TIP: To help you find the sound you are looking for, it is often necessary to create a Sound object (see Chapter 11) in your area and add sounds to this object, since by doing so you have access to a search function (i.e. if you type orc you will get all the orc sounds identified by the prefix or suffix used by Bioware to classify these sounds). Once you find the appropriate sound, write it down and remove the sound object. It will then only be a matter for you to open back your conversation and assign the sounds to the appropriate node.

- C** Journal: As for the “Journal” action, you can use it to add journal quest entries to the player’s journal when a specific node is spoken. The details on how to write the journal is covered in **Error! Reference source not found.** of this manual.

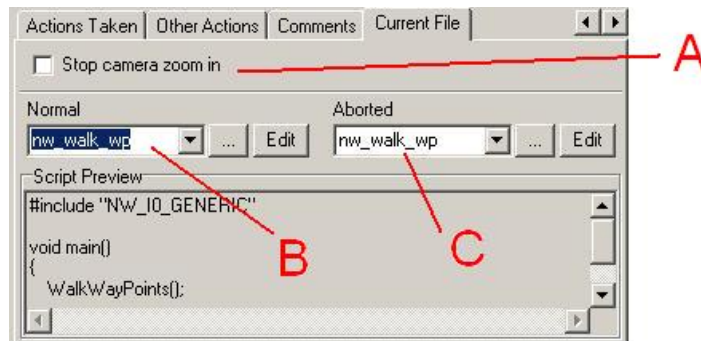
5.4.4 The Comments Property

Comments can be placed in the large blank rectangle appearing when you click on the “Comments” tab. These comments are intended to help the builder to keep track of any useful notes they may wish to leave on a node: they will not appear in the game.

5.4.5 The “Current File” Property of the Conversation File

Unlike the other properties described above in this Section, the “Current File” property applies to the whole conversation file and not only to a specific node.

Image 5.4.5 – Conversation Editor – Current File Tab



- A** If you check the “Stop camera zoom in” box, then the camera will not zoom on the owner of the conversation when the conversation is launched. Note that players have the ability to disable this function in their game options, so you might want to tick this box only when the conversation is fired during a cutscene where it would be totally inappropriate to have a zoom in and, otherwise, allow players to decide if they like the zoom in option on conversation or not.
- B** Here you can add a script that will run when the conversation finishes “normally”. “Normally” means that the conversation was spoken until a “END CONVERSATION” node was reached (i.e. the conversation was not aborted by the player by hitting the “ESC” key or by any other event like combat that could abort a conversation).

The default script that is automatically assigned here (“nw_walk_wp”) makes the owner of the conversation continue to walk his route if he has one.

- C** Here you can add a script that will run if the conversation is aborted. The conversation will abort if the player hits the “ESC” key, if he walks away from the owner of the conversation without finishing it, or if any other event (like combat) occurs which would make the player start a new action.

The default script that is automatically assigned here (“nw_walk_wp”) makes the owner of the conversation continue to walk his route if he has one.

5.5 Other Conversation Functions

There are a few other functions in the Conversation Editor that will not necessarily affect the conversation file itself, but can be used to facilitate writing the conversation tree.

5.5.1 Testing a Conversation

When conversation trees get big, it may become tedious to determine if you have written all branches completely and have made appropriate links for the conversation to appear to players properly.

The Conversation Editor provides a very basic function that will allow you to test your conversation and see how it turns out. To access this function, select the node where you want your test to start and right-click on it. Then select the “Test” option in the dropdown menu that appears.

This testing function is very, very basic. It will not make any of the script checks on conditional nodes. It should therefore only be used to get a first look at your conversation. The only way to really see how your conversation functions is to test your module in game.

5.5.2 Bookmarks

Bookmarks can be used in a lot of ways while writing larger conversation files:

- a) To keep track of nodes that start a new branch in the tree but that you have not yet had time to continue because you are writing another branch of the tree.
- b) To keep track of nodes that need to be rewritten.
- c) To identify the nodes which require scripts, while you do not want to write these scripts because you are in a writing inspiration boost.
- d) To keep track of any node, for any other reason you see fit.

The use of bookmarks really is to make a list of reminders of nodes that you want to access quickly without having to search for them at a later time.

To add a bookmark to a node, right-click on it and choose the “Bookmark” option in the dropdown menu that appears.

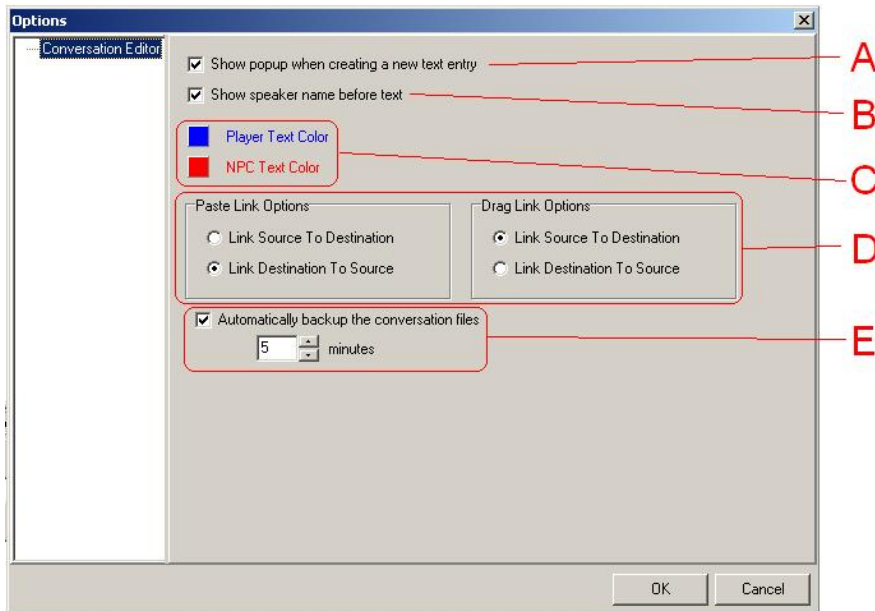
To access a node to which you have added a bookmark, click on the “Bookmarks” tab at the bottom left of the Conversation Editor screen. You will see a list of all nodes on which you have placed a bookmark. Double-click on the node in the bookmarks list that you want to access and the conversation tree will open and bring you to the selected node.

IMPORTANT NOTE: Bookmarks are not saved with the conversation file. Therefore, bookmarks will remain on nodes only until you close the conversation file.

5.5.3 Conversation Editor’s Options

The Conversation Editor’s Options can be accessed by clicking on the “grid-look-alike” icon button in the top button bar of the editor (as shown by the letter “B” on Image 5.2a above). The following screen will then appear:

Image 5.5.3a – Conversation Editor – Options



- A** When this box is enabled, adding a new node to a conversation will be made through a popup window. If it is disabled, then the new node will be added directly to the conversation tree.
- B** When this box is enabled, the tag of the speaker (or reference to the owner of the conversation) shows before the text to be spoken.
- C** Here you can change the colors of the nodes that are spoken by the player or by the owner of the conversation.

- D Changing these settings can be very tricky since they totally reverse the loop creation process (i.e. paste as a link) and the drag link process. It is not recommended to change anything in that area.
- E This is the setting for the auto-save functionality of the conversation files. When the toolset do not close normally, the backup conversation files can be found in the list of conversation files, named as "temp...".

5.6 *Running a Conversation*

Conversation files can only be run through a creature, a placeable object or a door object.

A conversation file assigned to a non-hostile creature will run simply by having the player clicking on the creature. However, for a conversation file assigned to a placeable or door object to fire when the player character clicks on it you first need to attach a script to this object.

If you want a conversation to fire when the player clicks on a placeable object, you must place the following script in the OnUsed script event of the object:

Script 5.6a – Making a placeable object execute its conversation file

```
void main()
{
object oObject = OBJECT_SELF;
object oPC = GetLastUsedBy();
AssignCommand(oPC, ActionStartConversation(oObject));
}
```

You must not forget to make your placeable object usable for this to work.

If you want a conversation to fire when a player clicks on a door, you must add the following script either in the OnAreaTransitionClick script event of the door (if you want the conversation to start when the door is opened and the player clicks on the area transition) or in the OnFailToOpen script event of the door (if you want the conversation to start when the door is locked and the player tries to open it):


```
void main()
{
object oPC = GetClickingObject();
object oDoor = OBJECT_SELF;
AssignCommand(oPC, ActionStartConversation(oDoor));
}
```

5.7 Designing Tips

Provide numerous answer options in conversation: Conversations do not necessarily have to be very long with a lot of multiple answers. However it is still important to give the player the possibility to pick between a few choices of answers, especially when the conversation is important to the story, in order for them to be able to roleplay their character. The multiple choices could all lead to the same result, but the player will have a more enjoyable playing experience than if he does not have any choice at all.

Make sure to properly verify the spelling in your conversations: Bad spelling and grammar is something that must be avoided at all costs. Conversations full of typos or that are not following the simplest rules of grammar will lead most players to stop playing the module. In order to review spelling of your conversation files, a very useful tool called "NWN Spellchecker" created by Hans Olsen is available to the building community on the NWWvault here:

<http://nwwvault.ign.com/Files/other/data/1056974990276.shtml>

You will need Microsoft Word to run this tool.

Give a conversation file to all NPCs: Every non-hostile NPC should have a conversation file. This conversation file could consists on a simple on-liner statement, but there are not many things that ruin the atmosphere in a module more than a city full of citizens that have nothing to say.

Use the "I have another question" loop so that players do not have to engage again the same conversation to be able to ask all questions that were available: It is a standard practice in a conversation file to group all generic questions under one branch. Then, when an answer is given to a question, it provides an "I have another question" answer option to the player. When it is selected, it brings the player back to the original list of questions. This technique should be used instead of simply ending the conversation after a first question was asked. Players do not generally like to have to engage in the conversation again to be able to ask further questions. But keep in mind when you use this technique that it is important to provide at the very least two questions to be asked that are not subject to a condition (a "Text Appear When..." condition). When players choose the "I have another question" option expect to be able to ask more questions. If all your questions are conditional, then you might upset some players that do not meet these conditions and only see the exact same question they just asked after clicking on the "I have another question" option.

5.7.1 Starting Conditional Scripts

There is a large number of starting conditional scripts created by BioWare that cover most of the common situations this is used for. Here is a short list:

nw_d2_arcane = is PC wizard, sorc, or bard?

nw_d2_chrh = does PC have high charisma?

nw_d2_classame = is PC same class as NPC speaking?

nw_d2_classbbn = is PC a Barbarian?

nw_d2_classholy = is PC a Cleric or Paladin?

nw_d2_classno = does none of the PC's classes match the NPC speaking?

nw_d2_divine = is PC a Cleric, Paladin, Druid, or Ranger?

nw_d2_evil = is the PC evil?

nw_d2_female = is the PC female?

nw_d2_ffemale = flirt check: NPC female and PC male with high charisma?

nw_d2_fmale = flirt check: NPC male and PC female with high charisma?

nw_d2_heal_0 = is the PC of a class with healing spells?

nw_d2_inth = does PC have high Intelligence?

nw_d2_male = is the PC male?

nw_d2_naked = is the PC naked? (75% chance of noticing)

nw_d2_nature = is the PC Druid or Ranger?

nw_d2_perseasy = easy persuade check

nw_d2_pershigh = hard persuade check

nw_d2_persmed = medium persuade check

nw_d2_racediff = is PC and NPC of different races?

nw_d2_racesame = is PC and NPC of same race?

nw_d2_shady = does PC have thief skills?

nw_d2_warrior = is PC a fighter, barbarian, paladin, or ranger?

nw_d2_wishigh = does the PC have high wisdom?

x0_d2_hascompan = does the PC have animal companion that the NPC can see?

x0_d2_hasfamil = does the PC have familiar that the NPC can see?

x0_d2_insight = does PC have high Int or Wis?

x0_d2_ischao = is the PC of chaotic alignment?

x0_d2_isgood = is the PC of good alignment?

x0_d2_islawf = is the PC of lawful alignment?

x0_d2_isneut = is the PC of neutral alignment?

x0_d2_skloreeasy = easy lore check

x0_d2_sklorehard = hard lore check

x0_d2_skloremed = medium lore check

x0_d2_skperfeasy = easy perform check

x0_d2_skperfhard = hard perform check

x0_d2_skperfmed = medium perform check

x0_d2_sктаuneasy = easy taunt check

x0_d2_sктаunhard = hard taunt check

x0_d2_sктаunmed = medium taunt check

x1_intimdt_easy = easy intimidate check

x1_intimdt_hard = hard intimidate check

x1_intimdt_med = medium intimidate check

x2_appraise_easy = easy appraise check

x2_appraise_hard = difficult appraise check.

x2_appraise_med = medium appraise check.

x2_bluff_easy = easy bluff check.

x2_bluff_hard = hard bluff check.

x2_bluff_med = medium bluff check.

xp1_less100hp = the NPC has less than 100% hitpoints?

xp1_pcisblind = is the PC blind?
xp1_pcisdiseased = is the PC diseased?
xp1_pcisdwarf = is the PC a dwarf?
xp1_pciself = is the PC an elf?
xp1_pciselven = is the PC elf or halfelf?
xp1_pcisgnome = is the PC a gnome?
xp1_pcishalforc = is the PC a halforc?
xp1_pcishuman = is the PC a human?

5.8 *How to...*

5.8.1 ... have a NPC remember a player from a previous conversation.

This can be done by creating local variables on the creature. Although this must be done through scripting, only a few scripts that can be used over and over again in any conversation, without having to create new scripts for each creatures, can be used.

The first script you will need is the following:

```
void main()
{
    string sName = GetName(GetPCSpeaker());
    SetLocalInt(OBJECT_SELF, sName+"talked", 1);
}
```

This script must be created once (we suggest you to name it "s_settalked1" for easy reference). Place it in the "Action Taken" property (see Sub-Section 5.4.2) of the conversation node that must be reached in the conversation by the player for you to consider that the creature will remember the player in future conversation. If you want the creature to remember the player up front the next time it engages in a conversation with it, put this script in the "Action Taken" property of the first node of the conversation.

Next you must create a second conversation node to the root of your conversation tree. This new node must be placed before the previous root node. To move a newly created node to the top of the tree, click on it, hold your mouse button, and drag it on top of the other node. Then, in the "Text Appears When..." window (see Sub-Section 5.4.1) of this new node, place the following script:

```
int StartingConditional()
{
    string sName = GetName(GetPCSpeaker());
    int nTest1 = GetLocalInt(OBJECT_SELF, sName+"talked");
    if(nTest1!=1) return FALSE;
    else return TRUE;
}
```

This script must also be created only once (we suggest you to name it "sc_talked1") and can be used in any conversation file. Write this node in a way that will show that the creature remembers the player that is engaging again in the conversation with him (i.e. something like: "You again? What do you want?").

5.8.2 ... have a henchman intervening in a conversation.

This can be done by using the method described in the Sub-Section 5.3.4 above, about creating multi-party conversations. To add the henchman as a possible speaker in your conversation, you need to click on the "Add" button as shown in this Sub-Section. You must however keep in mind that only creatures that have a unique blueprint can be added as a participant to a conversation. If your henchman was not created from a blueprint or if its tag is different from the blueprint you have used to create it, then you will need to make you henchman as a new blueprint. This can be done by following the comments made to that effect in Section 4.2. Note however that if the henchman is too far from the player when the conversation is running, then a generic portrait with a reference to an "Unknown Speaker" will rather be seen by players. You might therefore want to make sure (maybe by having the henchman jump near the players) that the henchman is near when such a conversation is run.

CHAPTER 6 Journal

The Journal is used by players to keep track of the quests that were entrusted to them and of important information they have found. Since most players will not want to take notes by themselves, the toolset provides you with the Journal Editor that permits you to write the basics of a player's quests journal and to reveal it to players while they progress through your module.

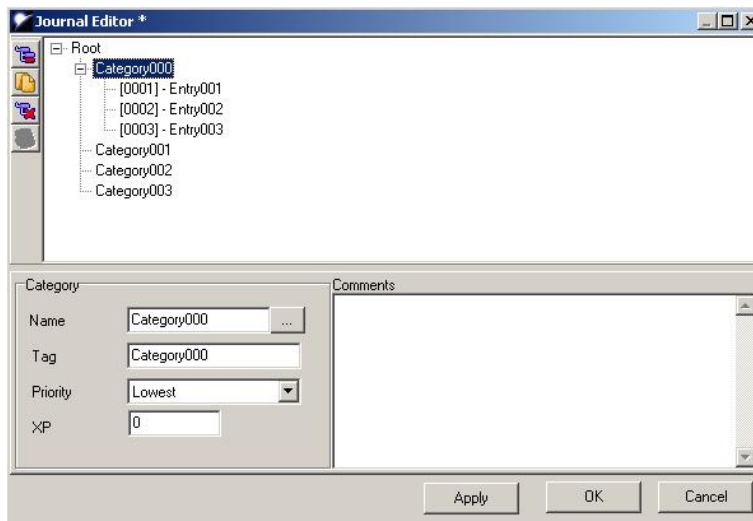
To access the Journal Editor, open the "Tool" menu of the toolset and choose the "Journal Editor" option.

6.1 Journal Categories

When you first open the Journal Editor, you will find that the Journal looks a lot like a conversation file. It starts with a root, to which will be attached all Journal Categories.

A Journal Category is a quest. To add a Category to the root of your module's journal, click on the first button at the top left corner of the Journal Editor.

Image Error! Reference source not found. – Journal Editor Screen – Journal Category



When you create a new category in your module's journal and you select it, you will obtain the properties shown in the image above for this specific category. These properties have the following purpose:

Name: As you know, when a player opens up his journal, he sees a list of quests that are still active and also those that are done. If he clicks on the title of the quest, he sees the details of that specific quest. The name of the category is in fact the title of the quest that will appear in the players' journal.

Tag: The tag of the category is used when you want to update the journal of the players through a script. Unlike in the other property windows (such as NPCs, placeables, etc) tags assigned to Journals are not automatically converted (all tags are normally converted to lower letters and any spaces between letters are removed). Make sure to only use lower letters and to not leave any spaces between letters when assigning tags to journals. Updating the journal for players is covered in Section **Error! Reference source not found.**

Priority: Players have the option in the game to organize their journal according to either the chronological order in which the quests were assigned or the priority level of each quest. For the second option to work for them, you must assign a priority level to each quest in your module represented by a journal category.

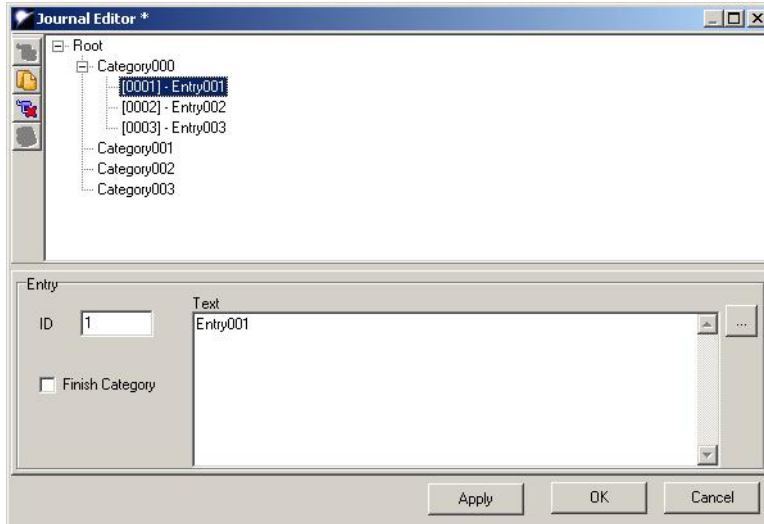
XP: Here you designate how many experience points to attribute to the quest represented by this journal category. **Note that these experience points will not be automatically given to players when they complete the quest.** This information is stored in the journal and must be accessed through a script if you want to give these experience points to players. The scripting command that must be used for that purpose is `GetJournalQuestExperience()`.

Comments: The comments are for your personal use only, they will never appear to players in game. You can write any notes in this box that may be useful to you to keep track of the use of this journal category.

6.2 Journal Entries

Once you have created a Journal Category, add the entries that are related to it. The entries are basically the text that players will see in their journal as the quest evolves further to their actions.

To add an entry to a journal category, select the relevant category and click on the button at the top left corner of the Journal Editor. New entries will then be added to the category, each having its own properties as shown in the following image:



ID: The “ID” is a numerical value that identifies the state of the quest. This number is used when you update the players’ journal to determine the text that should be revealed to them at that specific stage. Updating the journal for players is covered in Section **Error! Reference source not found.**

Finish Category Box: If you tick this box, the quest will be moved to the completed quest list in the players’ journal once they have reached this quest entry. A journal category may have more than one “finish category” entry, which is quite useful when there are many ways to end a quest and you want the wording appearing for that quest to take the many ending possibilities into consideration.

Text: The “Text” is what players will see in their journal when they click on the name of a quest (i.e. a journal category). The many entries of a journal category are there to provide the ability to update the journal to show the evolution of the instructions and information available to players as they evolve in the story of your module.

Note that players will see only the text that you have added for the entry that is currently in effect in their journal. They will not see the previous text entries. Therefore, if you want them to still see, as an example, the original text entry in which you provided all the details of the quest that was entrusted to them, you should copy and paste this original text in each one of the entries and add after it the text that is specific to the new entry.

6.3 Updating the Journal for Players

The players’ journal will not be updated in the game unless you tell the engine when players have accomplished the required elements either to have a new journal category opened for them or to pass from one journal entry to another one in an active journal category.

There are two ways to add or update an entry in the players’ journal: through a conversation or through a script.

6.3.1 Updating the Journal Through a Conversation

The properties of a conversation node, under the “Other Actions” tab, permit you to add/update a journal entry without any scripting. Details are in Sub-Section 5.4.3. To use this method:

- 1) Determine at which node in the conversation you want the journal to be updated and to select this node.
- 2) In the first dropdown menu appearing under the title “Journal” of the “Other Actions” properties of this node, select the journal category name that you want to update.
- 3) In the second dropdown menu appearing in the same place, select the journal entry ID that should appear in the players’ journal for this specific journal category (i.e. quest).

6.3.2 Updating the Journal Through a Script

The scripting command you are looking for to do this is: `AddJournalQuestEntry()`.

Yes, but where do I add this scripting command you ask? Well, it all depends on when you want the journal entry to be added or updated. If you want it to happen when a player has killed a creature, then place the command line in the `OnDeath` script event of the creature. If you want it to happen when a player has acquired an item, then place it in the `OnAcquireItem` script event of the module. As seen in the previous chapters of this manual, every object has its own script events in its properties so determine which one of these events should cause the journal to be updated.

6.4 Designing Tips

Do not provide more information in the journal than what players actually know: It is a common mistake to add “hints” in the journal to help players to complete a quest or to tell them what to do. A journal is not a horoscope, it should include only information that was provided to players. Hints should be given in the story, whether through conversations with NPCs or other means **and then** the journal should be updated with these hints, but not before the players have actually found them.

Write your module’s journal each time you build another quest element in your module: The journal is considered as an important element by players. It is a good habit to write the module’s journal as you introduce the plot in your module to avoid forgetting items of information that are quite clear when you have just written the quest element (especially in conversation). A journal written at the end of the module building process will generally look rushed and short of useful details.

If a quest requires players to go someplace, then write the directions in the journal for the players: Players usually hate to get lost. If the quest provider has given directions to players, you should put these directions in the journal. Keep in mind that players might not do the quest as soon as they receive it, so they will most probably not remember the directions that were given during a conversation.

CHAPTER 7 Items

Players may see items as a threat, but perhaps also as a reward for their hard work. A module will often come with a load of custom made items, and in this chapter we will provide you with the details on how to create items for your module.

7.1 Creating an Item Blueprint with the Item Wizard

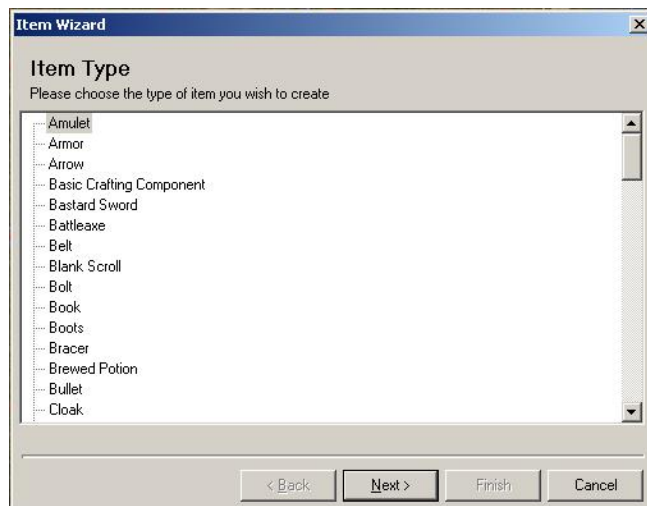
Creating an item with the Item Wizard is very easy, but unfortunately the end result will most probably disappoint you. A lot of editing of the item's properties is usually needed to really create the item you have in mind. Item editing fortunately is quite easy, but it is simply not done through a wizard. Section 7.3 will provide you with detailed comments on how to edit the properties of an item, but at this point let us see what the Item Wizard has to offer.

To launch the Item Wizard, you need to either:

- i) Choose the "Item Wizard" option in the "Wizard" menu of the toolset; or
- ii) type "Ctrl+Alt+I".

The wizard will then guide you through the four following steps to create an item object:

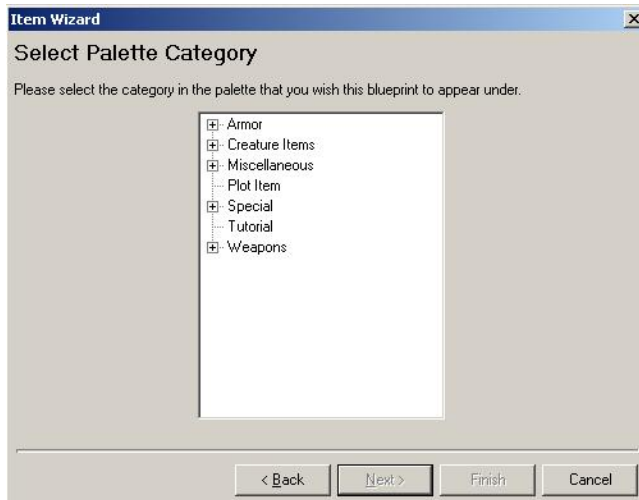
Image 7.1a – Item Wizard – Step 1



STEP 1: Choose here which type of item your new item will be. Notice that all the weapon and armor types are listed here, as are all the other types of items that can be equipped (rings, amulets, etc.). If your intention is to create an item that is not a piece of gear (a letter, a monster's head, a crystal or any other usually quest-related item), you must select one of the "Miscellaneous" types.

Once the item is created, you cannot change its type. If you are not happy with the type you selected for an item, then you must delete it and start all over again.

Image 7.1b – Item Wizard – Step 2



STEP 2: Next determine in which category of the custom item palette your new item blueprint should appear.

If you have CEP installed on your module, you will notice that the custom item palette is full of pre-generated items that were created by the CEP team. If you do not want your own items to be mixed up with those coming with CEP, use the "Special" category to store your item blueprints.

Image 7.1c – Item Wizard – Step 3

The screenshot shows a dialog box titled "Item Wizard" with a close button in the top right corner. The main heading is "Name" with the instruction "Please enter a name for the new item." Below this is a text input field labeled "Name:". A checkbox labeled "Magical" is present. There are two radio button groups: "Item Level" with options "1-5", "6-10", "11-15", and "16-20"; and "Item Quality" with options "Low", "Medium", "High", and "Godly". At the bottom are buttons for "< Back", "Next >", "Finish", and "Cancel".

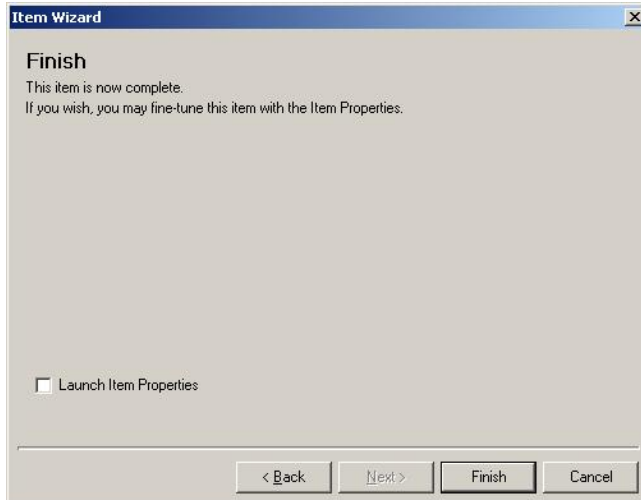
STEP 3: Two things can be done during this third step. First, choose the name for your item.

Second, tick the “Magical” box if you want to create an item with magical properties. If you do not tick this box, it does not mean that you will not be able to assign magical properties to your item later on when you will be editing it. The “Magical” box in fact generates random magical properties for your item. Therefore, if you name your new item “Fire Sword of Hell” and you tick the “Magical” box, the properties assigned to your item by the engine might be extra frost damage points made on hit, or some other unrelated random effect.

You can therefore see that using this random magical properties tool has very limited use. Most of the time you will find yourself removing the random properties that were assigned to your item by this tool. It is better to assign magical properties to your item by editing its properties directly (see Section 7.3).

The “Item Level” and “Item Quality” properties provided in this screen are only a complement to the random magical item properties selector tool. By the “Item Level” property you determine the item level restriction range for the item. The “Item Quality” determines where within this range the item level restriction range should be located. As an example, if you choose “11-15” in the “Item Level” property, then selecting “Low Item Quality” will most probably generate random magical properties that will make the item restricted for players of level 11 and above, while selecting “Godly Item Quality” will most probably make it for player characters of level 15 and above.

Image 7.1d – Item Wizard – Step 4



STEP 4: Well, there is nothing much to be done here. By clicking on the “Finish” button, the blueprint of your newly made item will be added to the custom item palette in the category you chose under step 2 above.

If you want to edit the item properties at this stage (editing can be done at any time), tick the box to that effect and the Item Properties Screen will open. Editing the properties of an item is covered in Section 7.3 below.

7.2 Creating an Item blueprint from an Existing Blueprint

You can also create a new item blueprint by using an already existing one as a model. In order to do so, right-click on the item blueprint in the palette that you want to use as a model and select the “Edit Copy” option in the dropdown menu. At that point, the Item Properties Screen will open so that you can customize this item at will. Note that you can also create a blueprint from an item already in the game, by choosing “Add to Palette” from the item’s dropdown box.

7.3 The Properties of an Item

The properties of an item object are easy to understand and to modify. In fact, many properties cannot be edited since they are hard coded within the game and reflect the core D&D rules. As an example, a longsword will cause 1d8 of damage on a successful hit. Although you can add magical properties that increase the amount of damage made, there is no possible way to change this “1d8” basic damage roll to any other dice roll.

To modify the properties of an item:

- a) if you want to edit the properties of all items that will be generated using this blueprint, right-click on the blueprint and choose the “Edit” option in the dropdown menu; or
- b) if you only want to edit the properties of a specific item and not of the blueprint used to generate it, right-click on the item itself in the inventory of the creature or placeable object holding the item.

The Item Properties Screen will then open. The properties of the item are grouped under the following 5 categories:

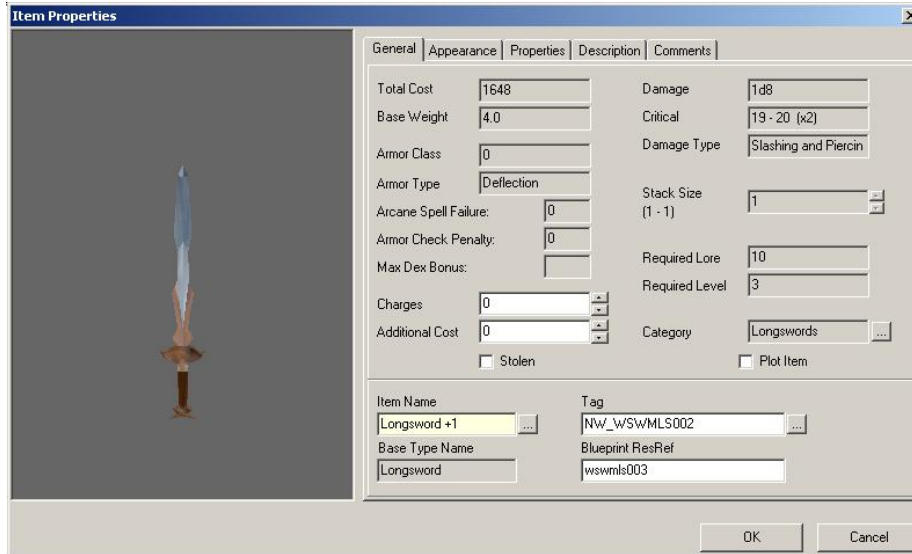
- i) The General Item Properties (see Sub-Section 7.3.1)
- ii) The Appearance of an Item (see Sub-Section 7.3.2)
- iii) The Magical Item Properties (see Sub-Section 7.3.3)
- iv) The Description of an Item (see Sub-Section 7.3.4)
- v) The Comments Property of an Item (see Sub-Section 7.3.5).

IMPORTANT NOTE: When you edit the properties of an item blueprint, items that you have already placed in the inventory of creatures and placeable objects (like containers or chests) placed in your module will not automatically be updated. If you want all previously placed items to be updated in accordance with a change you have made to the item blueprint, right-click on the blueprint and choose the “Update Instances” option in the dropdown menu.

7.3.1 The General Item Properties

This screen provides you with all the statistics about your item that will have an effect in game. Many of these statistics are applicable to only one type of item and most of them cannot be edited since they reflect the hard core rules of D&D about items (mainly the attributes of weapons and suits of armor).

Image 7.3.1 –Item Properties Screen – General Tab



Total Cost: This is the actual value, in gold pieces, of your item. You must keep in mind that merchants in the game may have pricing mark-up or mark-down strategies (depending on whether they buy or sell the item) that will adjust the price of the item. (See Sub-Section 14.3.1 for more details.)

The value of the item is set automatically based on the base type of the item and its magical properties. There is no way to reduce the value of an item. If however you want to increase it, you can modify its “Additional Cost” property.

Base Weight: The base weight of the item is also set automatically based on the base type of the item and, for armor, depending on its torso appearance.

The only way to modify the weight of an item is by adding either the “Base Item Weight Reduction” or “Weight Increase” properties to the item. (Note however that these properties are not available for all categories of item.) Keep in mind that when you add the “Base Item Weight Reduction” property to your item, your item is then considered to be magical and therefore its cost and level restriction will increase, often significantly.

Armor Class: The base armor class is a statistic that will apply only to suits of armor. This statistic is taken into consideration when determining whether a blow made by an opponent will hit the wearer of the suit or not.

The base armor class of an armor suit is strictly based on its torso appearance. If its torso appearance is made of leather, then the base armor class will be set to the one provided in the standard D&D rules for leather armor. It is impossible for you to create a leather armor that will have the same base armor class property that of a full plate armor. If you want to modify the armor class of an armor item, you will need to add either an “AC Bonus” or “Decrease AC” property. Keep in mind that when you add the “AC Bonus” property to your item, your item will be considered magical

and therefore its cost and level restriction will increase. (This does not happen with the “Decrease AC” property.)

Armor Type: When your item is a weapon, the “Armor Type” property will indicate what type of armor modifier will be affected when its wearer is hit by a creature using the weapon. This property cannot be changed.

As an example for how this property may have an impact, the longsword shown in Image 7.3.1 will affect armor with properties related to “Deflection” armor type. Therefore, if a suite of armor has the “Decrease AC – Deflection Modifier”, then the modifier will be taken into consideration in the calculation that determines if the wielder of the longsword will hit the wearer of the armor in a combat.

Arcane Spell Failure: This statistic applies only to suits of armor and will determine, in percentage, the chance that its wearer will fail to cast arcane spells. This statistic is determined by the category type of the suit of armor (which is based on its torso appearance), based on the D&D rules. This statistic can be modified by adding and editing the “Arcane Spell Failure” property of your item.

Armor Check Penalty: The armor check penalty modifier is applicable only to suits of armor and is determined by the category type of the suit of armor (which is based on its torso appearance), based on the D&D rules. It will affect all dexterity based skill checks made by the wearer of the armor. The Armor Check Penalty modifier cannot be modified.

Max Dex Bonus: The maximum dexterity bonus property of an armor suite will limit the dexterity bonus of its wearer. The dexterity bonus is used to determine the dexterity based armor class modifier of a creature (see Sub-Section 4.3.2) and also affects all roles made on dexterity based skill checks.

The Max Dex Bonus is determined by the category type of the suit of armor (which is based on its torso appearance), based on the D&D rules. It is not possible to change this property.

Charges: Here you can set the number of charges allocated to your item. Items with spell casting properties can be based on charges. As an example, let us say that you have added the “Cast Spell – Acid Splash” property to your sword. By editing this property, you can set its parameters so that it costs 3 charges per use to make use of this property.

Additional Cost: This property is used if you want to increase the value, in gold pieces, of your item.

Damage: The base damage roll of a weapon is used to determine how much damage its wielder will cause by a successful hit in combat. This property is based on the base type of the weapon, in accordance with the D&D core rules. It is impossible to change the damage roll of a weapon, although you can add or reduce the amount of damage made by a weapon by giving it the “Damage Bonus” or “Damage Reduction” property.

Critical: In the D&D rules for combat, which are implemented in NWN, a d20 die is rolled to determine if a creature is successful in an attack attempt. This d20 roll can be adjusted by many factors in the game. The “Critical” property of a weapon determines what the wielder of the weapon must get on the unmodified d20 attack roll to cause critical damage to his opponent. The “Critical” property is based on the base type of the weapon, in accordance with the D&D core rules, and cannot be modified.

Damage Type: Many suits of armor or creature skins will be more or less effective versus certain damage types. The damage type of a weapon item determines which type of damage the weapon will make on a successful hit. This property is based on the base type of the weapon, in accordance with the D&D core rules, and cannot be modified.

Stack Size: Some items can be stacked, like potions and ammunitions, so that they do not occupy too much space in their holder's inventory. By modifying the "Stack Size" property of an item, you adjust its quantity.

Required Lore: This is the required lore skill level that is needed for a player character to be able to identify an item. This property is based on the base total cost of the item, in accordance with the D&D core rules, and cannot be modified. If you do not want your magical item to be unidentified, then you need to tick the "Identified" box in the "Properties" tab of your item's properties.

Required Level: This value determines the minimum level that a character needs to be to use the item. This value will only have an effect if the item level restriction rule is enabled for your server. (See Sub-Section 7.5.1 for greater details.) This value is based on the total cost of the item, in accordance with the NWN rules, and cannot be modified.

Category: The "Category" value informs you where the item blueprint is stored in the custom item palette. You can change the category by clicking on the "..." button.

Stolen Flag: By ticking the stolen flag, you will set the item as being stolen. Some merchants may not buy stolen goods or, if they do, they may buy them at a price well under the actual cost of the item. (See Chapter 14 for more details on the properties of a merchant.)

Plot Flag: By ticking the plot flag, you identify the item as being part of the plot. A plot item cannot be sold or destroyed.

Item Name: The name of the item can be changed in the General properties of the item. By clicking on the "..." button next to the name of the item, you can set the name of the item depending on the language version of the game played.

Tag: The tag of the item permits you to identify the item in scripts.

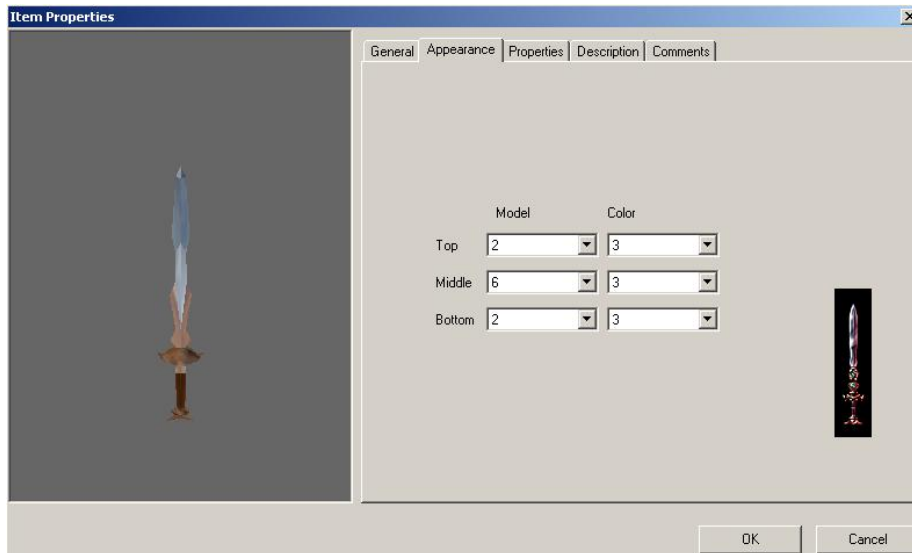
Base Type Name: The base type of the item is selected in the item selection process. (See Sub-Section 7.1.) This property will determine all the statistics related to the item, in accordance with the core rules of D&D. The base type of an item cannot be changed. If you are unsatisfied with the base type of a specific item, there is nothing else for you to do but delete the whole item blueprint and create a new one.

Blueprint ResRef: The blueprint resref is assigned during the item creation process and is the unique reference identifying the item. This reference is used when you wish to create an item through a script. It is also useful when you want to update all items already placed in your module in accordance with a modification you have made to the blueprint that was used to create them (this can be done by right-clicking on the blueprint and selecting the "Update Instances" in the dropdown menu).

7.3.2 The Appearance of an item

Every item type will provide you with different options to set the appearance of your item. You will find in the image below those that are available for a longsword.

Image 7.3.2 – Item Properties Screen – Appearance Tab



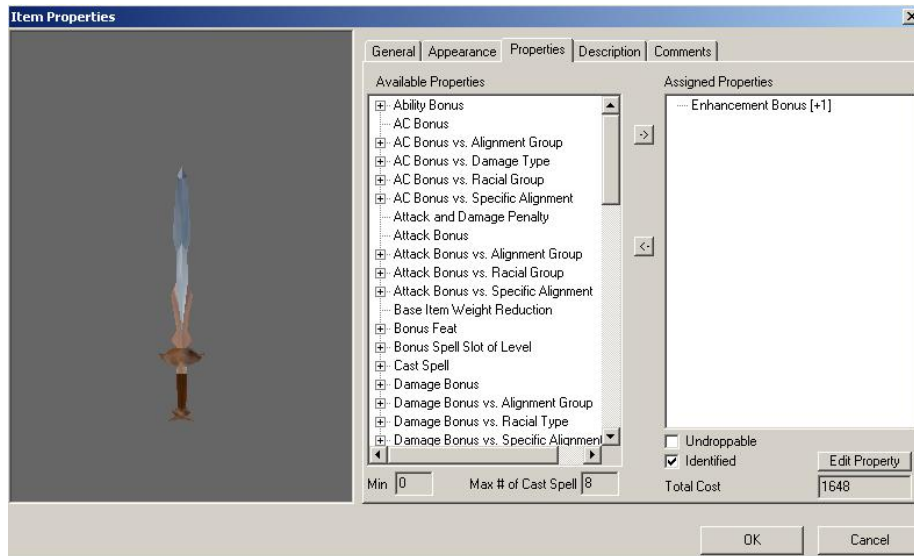
For all items but the suits of armor, the appearance properties are strictly aesthetic. However, for a suit of armor, you must keep in mind that the base armor class of your armor (and therefore many other statistics/properties that are based on the base armor class) is set depending on the appearance of the torso of the armor.

7.3.3 The Magical Item Properties

Now this is where the true weapon editing is being done. Item properties will (except for a few exceptions) make the item magical by giving supernatural attributes to its wielder or by generating devastating effects when used.

When you select the "Properties" tab of the Item Properties screen, you will see the following lists:

Image 7.3.3 – Item Properties Screen – Properties Tab



Available/Assigned Properties Lists: The Available Properties will differ depending on the type of item you are editing. To add item properties, select the property and click on the right arrow button to insert it into the Assigned Properties list. If you want to remove a property, select it from the Assigned Properties list and click on the left arrow button. Alternatively, you can simply double-click on any item property to add or remove it from the assigned properties list

Edit Property Button: The “Edit Property” button is used to adjust any property that is already assigned to the item. As an example, the longsword in the image above is a +1 weapon. If you want to make it a longsword +2, select the “assigned property” you wish to modify and click on the “Edit Property” button. A window with the available parameters will appear. The same process would have to be followed if you have assigned a “Cast Spell” property to your item and you want to determine how many times per day you want its holder to be able to use it or how many charges are required to use it.

Undroppable Box: If you tick the “Undroppable” box, then the item cannot under any circumstances leave the inventory of its holder. (The holder cannot sell it, drop it even on death, give it to another player or NPC, etc.)

Identified Box: If you tick the “Identified” box, the object will be identified when it is found by a player.

Max # of Cast Spell / Total Cost: The “Max # of Cast Spell” and “Total Cost” values are at the bottom of this screen only for reference purposes.

Most item properties’ names are self explanatory. A few of them however may need more explanation:

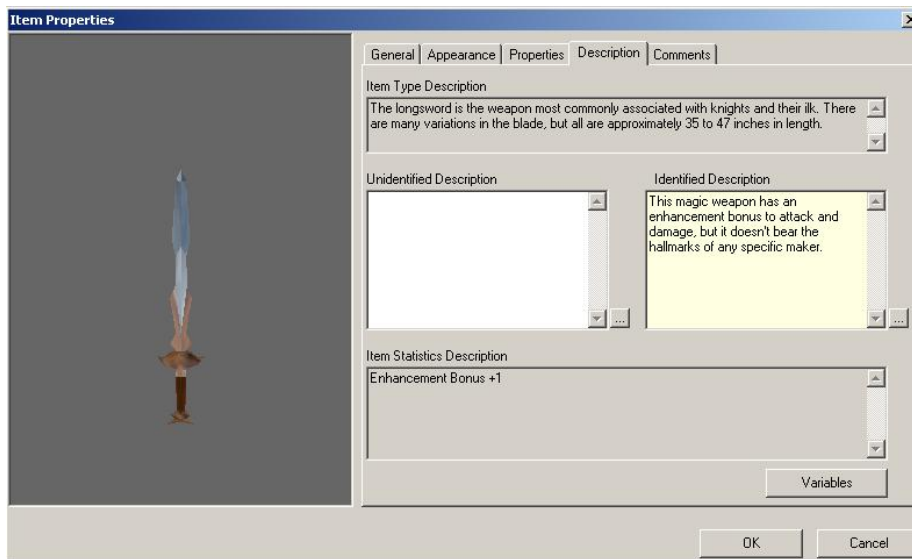
“Cast Spell – Unique Power / Unique Power Self Only” property: This item property has a lot of potential to create custom items. By adding this property to an item (any type of item), players that right-click on an item with this property will be given the opportunity to activate it from the radial menu appears. The effect of activating the item is whatever you design in the associated script. The script must be placed in the OnActivateItem event of your module. (See Sub-Section 2.2.2.)

“On Hit Cast Spell” property: Although this property may seem very cool, you must keep in mind that it is also very demanding on an engine. This property will cause the selected spell to be cast each time a successful hit is made with the item. Spells usually have visual effects and, in high level adventures, there may be many hits per round. Use with great care. Also note that this property will not fire if the weapon is used against a placeable object.

7.3.4 The Description of an Item

The “Description” of the item is the information that will be provided to players that examine it in game.

Image 7.3.4 – Item Properties Screen – Description



The “Item Type Description” is the description that appears to players if no others are provided in the “Unidentified Description” or “Identified Description” text box.

The “Unidentified Description” and “Identified Description” are the descriptions that players will see when they examine the item, depending on if they have successfully identified it or not. Many builders use the “Identified Description” to provide the history of the weapon. Since the item properties will be automatically listed by the game at the end of your textual description, you should not list them yourself.

The “Item Statistics Description” is the list of properties that were given to the item for reference purposes to help you to write your descriptions. This list is automatically added to the “Identified Description” that will be seen by players in game.

The “Variables” assigned to the item by clicking on the button to that effect have nothing to do with its description. These variables are rather used in scripts in which the item is involved. To assign a variable to an item that will be used in a script, click on the “Variables” button and the variables screen will open.

7.3.5 The Comments Property of an Item

The empty text box that appears when you click on the “Comments” tab of the Item Properties Screen is for you to use as you decide. These comments will not affect the item nor will they be shown to players. They are usually used to leave notes or a to-do list for the specific item.

7.4 Designing Tips

Refrain from giving hugely powerful items to players early on in the module: When you build your module, you must keep in mind that obtaining a powerful weapon is a very good motivation to use in designing quests. Players usually like powerful weapons, and obtaining them as a reward is a very good incentive. If a player can obtain a very cool looking and efficient sword with a blasting fire visual effect on it early in your module, you do not leave you much room to offer a better and cooler weapon to your players at a later stage in the module.

Do not put visual effects on all your custom made weapons: There is no doubt about it, weapons with visual effects look cool. You might however want to refrain from using them too often, especially on weapons that will be wielded by common creatures in your module. These visual effects are known to be demanding on your engine. In addition, if too many weapons have these effects, it detracts from the appeal of all weapons with visual effects, making them seem common.

Do not forget to mark an item as “identified” if players know what this item is: By default, all magical items that you create will not be marked as “identified” and players will therefore need to either make a successful lore check to identify it, use an identify spell on it, or pay to have it identified at a store. This can really be a pain for players when they know exactly what the item is because a NPC has told them in a conversation. To mark a magical item as identified, simply tick the box to that effect under the “Properties” tab of the item’s properties. This can also be done in a script, which is a more likely way to do it if an NPC has just described the item for you.

Only known items should be marked as stolen: The main reason of activating the stolen flag of an item is that merchants may not buy stolen items or, if they do, they may do it at a lower price. But how can a merchant know that a particular leather armor was stolen? Maybe adventurers write their name on their gear, but we doubt it is the case here. Only well known items in a region (like for instance a painting or maybe a custom weapon or armor that is known to be owned by a nobleman) should be flagged as being stolen, if it fits with the circumstances under which these items have been acquired by players.

7.5 How to...

7.5.1 ...implement or remove the item level restriction rule on my module

The mandatory item level restriction rule for solo modules has been removed since the release of HotU and it is now impossible to have a module flagged as using the item level restriction rule. In fact, the application of this rule is decided on the player's side, in the player's "nwnplayer.ini" file. Therefore, if your intention is to limit players from using gear before they reach a certain level in a solo module, the only way is to add a script in the OnPlayerEquipItem event of your module that looks something like this:

```
void main()
{
object oPC=GetPCItemLastEquippedBy();
object oItem=GetPCItemLastEquipped();
int nILR = GetLocalInt(oItem, "ilr");
int nClassPC1 = GetClassByPosition(1, oPC);
int nClassPC2 = GetClassByPosition(2, oPC);
int nClassPC3 = GetClassByPosition(3, oPC);
int nClassPC = nClassPC1 + nClassPC2 + nClassPC3;
//Do not fire the script if no item level restriction was assigned
//to this item
if(nILR==0) return;
//Do nothing if the player's level is equal or
//higher than the item level restriction
if(nClassPC >= nILR) return;
else
{
    SendMessageToPC(oPC, "You cannot use this item at your current level.");
    AssignCommand(oPC, ClearAllActions());
    AssignCommand(oPC, ActionUnequipItem(oItem));
}
}
```

The above script assumes that you have added an integer variable called "ilr" that is set to the level number at which you want players to be able to start to use the item to items that you want to be level restricted. (See Sub-Section 7.3.4 about adding variables to an item.)

In a multiplayer environment, the activation of the item level restriction rule is decided by the host when launching the server.

CHAPTER 8 Placeables

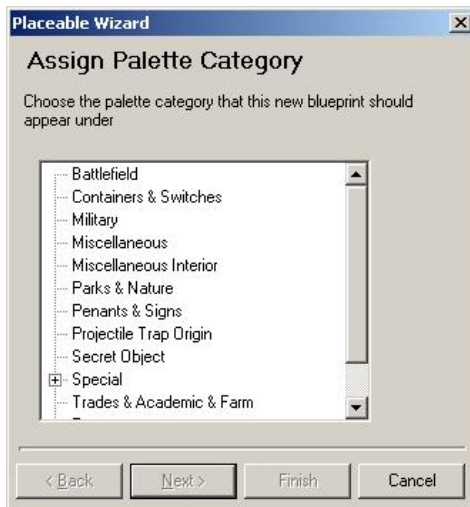
Placeable objects are not only used to garnish your areas, but they also can play an important role in the plot of your module. What is a dungeon without a lever that does crazy stuff? What is a mage tower without an orb in which players hesitate to look? In this chapter we will provide you with detailed comments on the properties of placeable objects and how to edit them.

8.1 Creating a Placeable Blueprint with the Placeable Wizard

You will find that there is not much need for you to create your own placeable objects. The Standard placeable blueprints palette (and the Custom one if you have CEP installed on your module) is well supplied with such objects, and all the possible appearance types for placeables are listed somewhere in the palette.

If for some reason you still feel the need to create a new placeable object blueprint, the Wizard might be the tool you are looking for.

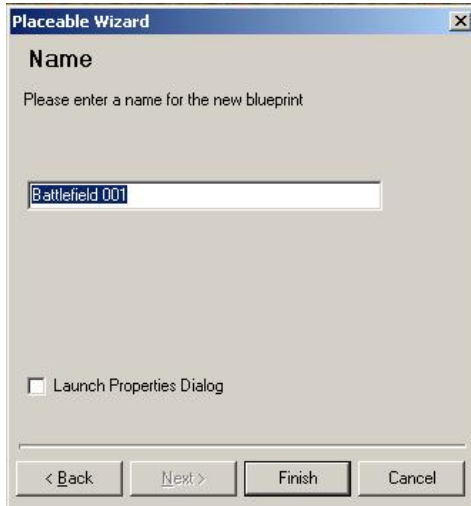
Image 8.1a – Placeable Wizard – Step 1



STEP 1: The first step of the placeable object creation process is to choose in which category you want the blueprint of your new placeable object to be archived.

If you have the CEP hakpacks installed on your module, you might want to place your creation in the "Special" category since the other ones are full of blueprints that come with CEP. Otherwise, you might lose a lot of time searching for your own creations.

Image 8.1b – Placeable Wizard – Step 2



STEP 2: For the second (and last) step of the placeable object creation process, you choose a new name for your new blueprint and... that's it.

You can of course tick the “Launch Properties Dialog” box if you want to edit the properties of you placeable object. You will find detailed comments about editing these properties in Section 8.3 below.

8.2 Creating a Placeable Blueprint from an Existing Item

The other way to create a new blueprint for a placeable object is to create a copy of an existing one and customize it to fit your own purposes.

In order to do so, right-click on the placeable in the palette you want to use as a model, and select the “Edit Copy” option from the dropdown menu. Another method is to add the placeable you will use as a template to an area, edit its properties and, once you are finished, right-click on it and select the “Add To Palette” option to make it a blueprint in the palette.

8.3 The Properties of a Placeable Object

Since the creation wizard does not help much in customizing your placeable object, it is by editing its properties that you will really make your object fit its purpose in your module.

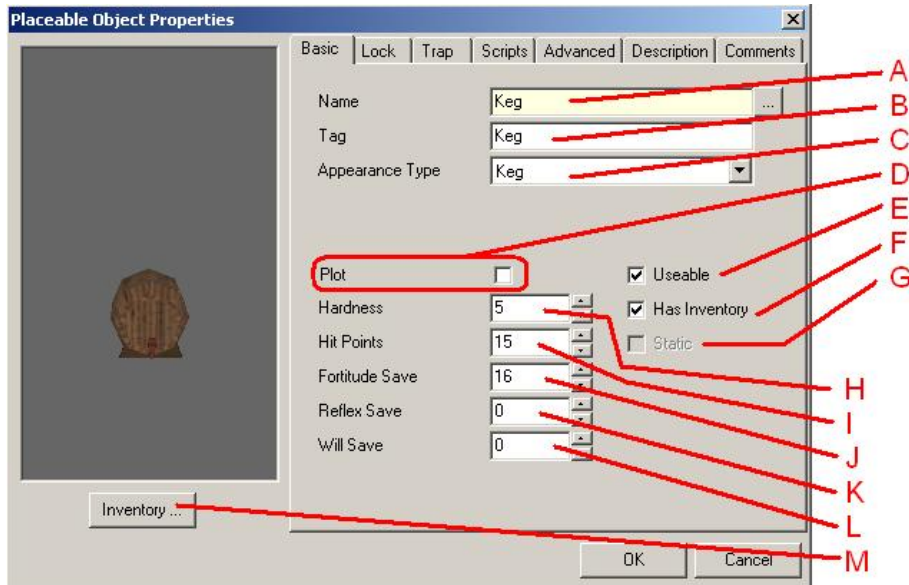
You can access the properties of a placeable object by:

- i) right-clicking on the placeable object and selecting the “Properties” option; or
- ii) selecting the placeable object and, in the “Edit” menu of the toolset, selecting the “Placeable Properties” option.

8.3.1 The Basic Properties of a Placeable Object

The following screen will appear when you launch the property editor for a placeable object:

Image 8.3.1 – Placeable Object Properties – Basic Tab



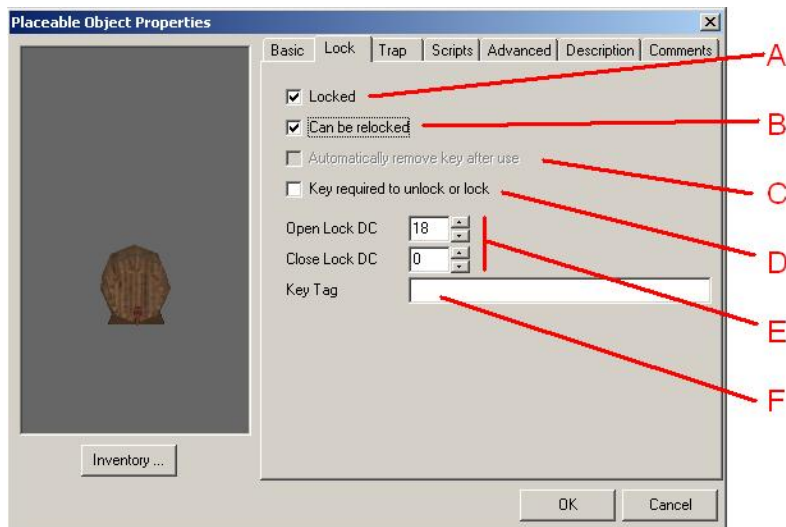
- A** Name: The name of your placeable object is for you to choose. By clicking on the “...” button next to it, you can edit the name that will appear to players using a foreign game version of NWN.
- B** Tag: The tag of your placeable object is mainly useful to identify this object in a script.
- C** Appearance Type: You can give any appearance type to your placeable object. This property is mainly aesthetic and will not affect the other properties of your object.
- D** Plot: If you tick the “Plot” box, the placeable object cannot be destroyed. Do not forget that placeable objects can not only be destroyed by players bashing them, but also by area of effect spells.
- E** Useable: If you tick the “Useable” box, your placeable object will be selectable by players in the game. Players clicking on the object will then fire the OnUsed scripting event of your object. (See Sub-Section 8.3.4 below.)
- F** Has Inventory: If you tick the “Has Inventory” box (which can be done only if you have first made your object “Usable”), then your placeable object will act as a container. The object then becomes selectable for players in the game and any player clicking on it will open its inventory, assuming it is not locked or in some other manner not easily opened.
- G** Static: By ticking the “Static” box, your placeable becomes part of the tile where it is located and no longer interacts in the module in any way. It is important to understand that scripts assigned to a static placeable object will not run. It is strongly recommended to have all placeable objects that were placed in an area only for an aesthetic purpose to be made “static” to limit the usage of your engine memory.

- H** Hardness: The hardness value tells you how many damage points the placeable absorbs on a successful hit without being affected. As an example, if the placeable object has a value of 5 to its hardness, then a player will need a weapon that does at least 1d6 of damage to be able to bash it (and will do 1 point of damage on rolls of 6 only). Any weaker weapons will be ineffective.
- I** Hit Points: This value is the same as for any creature or player character. It determines how many damage points you have to inflict to the placeable object before you destroy it.
- J** Fortitude Save: A placeable object can be the target of a spell. The fortitude save will therefore determines how well the placeable object will resist a spell that allows for a fortitude save.
- K** Reflex Save: The reflex save determines how well the placeable object will resist a spell that allows for a reflex save.
- L** Will Save: A The will save determines how well the placeable object will resist a spell that allows for a will save.
- M** Inventory: Any placeable object can act as a container. In order to activate this option, you must first tick the “Has Inventory” box. Then, when you click on the “Inventory” button, you can add any item you wish to the placeable object’s inventory.

8.3.2 The Lock of a Placeable Object

If you have ticked the “Usable” box in the Basic properties of your placeable object, then the “Lock” tab will appear at the top of the Placeable Object Properties screen. By clicking on this tab you will see the following:

Image 8.3.2 – Placeable Object Properties Screen – Lock Tab



- A** Locked: Ticking this box will confirm that the placeable object is locked and therefore players cannot have access to its inventory (or the OnUsed event will not run if the inventory option has not been activated on the placeable object) without first either unlocking it or bashing it.

- B** Can be relocked: Ticking this box will confirm that the placeable can be relocked. It is always possible to relock a placeable object through a script, you do not have to tick this box to make it possible. However, if you want players to be able to relock it (which makes more sense in permanent world modules than in solo player modules), this functionality is what you are looking for.

- C** Automatically remove key after use: If you have ticked the “Key required to unlock or lock” option, then you might want to consider activating this option as well if (i) players only have to unlock the placeable object once; and (ii) the key that is required has no other use than to open this placeable. Although it does not make much sense to have a key vanish once the player uses it, it does help them to keep their inventory free of items/keys that are no longer relevant.

- D** Key required to unlock or lock: By ticking this box, it will no longer be possible for players to unlock the placeable object and therefore to have access to its inventory unless the players first find the required key. You will then need to place the tag of the item that will permit players to open the placeable object in the “Key Tag” parameter.

Keep in mind however that ticking this box will not prevent players from bashing the placeable object. If you want to make it invulnerable to bashing, then you must tick the “Plot” box which appears on the Basic properties of the object.

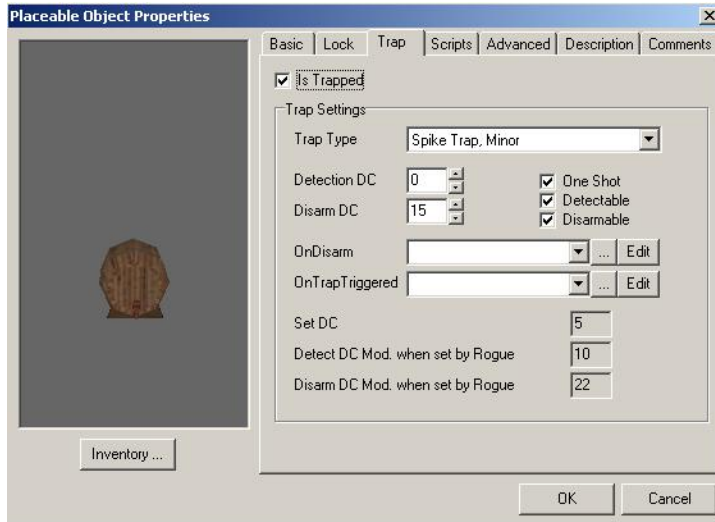
- E** Open/Close Lock DC: These parameters determine the level of success, using a d20 die roll, required to successfully unlock or lock the placeable object when a key is not required to unlock the object.

- F** Key Tag: Here you add the tag of the item that players must have in their inventory to be allowed to unlock the object when the “Key required to unlock or lock” option is activated. The item does not have to be a key; it can be a sword, a wand or any other item type. Keep in mind that if the “Automatically remove key after use” option is activated, the item identified here will vanish when the placeable is first opened, which might be a dirty trick to do to your players if the key is a sword.

8.3.3 The Trap of a Placeable Object

If you have activated the “Useable” option in the Basic properties of the placeable object, you will also have the possibility to place a trap on it. The trap will be triggered by any player that clicks on the placeable object. Here are the options related to the settings of a trap on a placeable object:

Image 8.3.3 – Placeable Object Properties Screen – Trap Tab



Is Trapped: Tick this box if you want a trap to be added on your placeable object. All the other parameters below will then become editable.

Trap Type: Choose here the type and the strength of the trap that you wish to set on your placeable object.

Detection/Disarm DC: The detection/disarm DC establishes the roll that must be obtained, on a d20 die, to detect/disarm the trap. Of course, spot and disarm skills will be applied to this check.

One Shot: If this box is ticked, the trap will fire only once. Otherwise, the trap will fire each time someone tries to use the placeable object, until the trap is disarmed. Keep in mind that this can be very nasty for non-rogue characters.

Detectable: If this box is ticked, the trap is detectable (upon a successful spot check on the Detection DC set for this trap). Otherwise, the trap will fire when the placeable object is used by a player without any indication that the trap was there.

Disarmable: If this box is ticked, then the trap is disarmable (upon a successful disarm check on the Disarm DC set for this trap). Otherwise, players will not be able to use the placeable object without triggering the trap.

OnDisarm: Disarming a trap will simply remove it from the placeable object. If you want other things to happen when players attempt to disarm the trap, then you can add a script here to that effect.

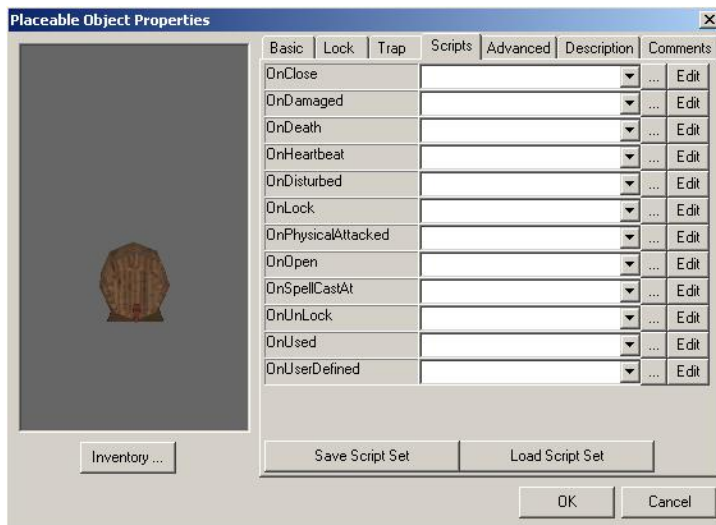
OnTrapTriggered: Triggering a trap will simply apply the appropriate damage to the player who has triggered it and make the relevant visual effect. If you want other events to happen when players trigger the trap, you can add a script here to that effect.

Set DC / Detect DC / Disarm DC mod when set by a rogue: These parameters are those provided in the core rules of D&D. Although they do not affect the properties of the trap you have placed on your placeable object, they can be used for guidance in adjusting the Detection and Disarm DC of your own trap.

8.3.4 The Scripts of a Placeable Object

Adding scripts to your placeable object will complement the other functions that are available in its properties. The events on which you can add a script are the following (the comments made in regard to each of these events have been provided by the invaluable Lexicon):

Image 8.3.4 – Placeable Object Properties – Scripts Tab



OnClose: This script fires when the placeable object gets closed. It implies that the placeable object's HasInventory function is enabled since otherwise there is no way to open the placeable object. This event will generally be used to make something happen when players have touched the placeable object (shut off the lights, lock a door, make a guard hostile, etc.).

OnDamaged: The script attached to this event fires when the placeable object is damaged.

OnDeath: The script attached to this event fires when the placeable object is destroyed for any reason (whether because it is bashed, affected by a spell, etc.). DestroyObject() will not trigger this event, although DM kill will (with an invalid last killer).

Scripts that execute as a result of this event should have their actions take effect immediately. Using DelayCommand() will not work. However, AssignCommand() could be used to place an instruction in the action queue of another object, for example an invisible object placeable or the module itself. Also note that if DestroyObject(OBJECT_SELF) is used in the OnDeath event, it will also cause associated UserDefined events, ie the Death event, to not execute at all.

OnHeartbeat: The script associated with this event fires every six seconds. This is good for a script that needs to check for something repeatedly or perform an action regularly. Be aware that too many OnHeartbeat scripts can be processor costly. Use other types of event scripts when possible.

OnDisturbed: The script attached to this event fires when the inventory of the placeable object is changed in any way (i.e. if a new item is added to its inventory or if an item is taken from its inventory). This of course implies that the placeable object's HasInventory function is enabled since otherwise there is no way to disturb its inventory.

This event also fires when you give gold through a script to the Placeable object, even though the gold can never be retrieved since it is not considered as an item.

OnLock: This script fires when the placeable object is locked. Of course, you will first need to enable that function in the "Lock" properties of the object.

OnPhysicalAttacked: The script attached to this event fires each time the placeable object is hit (it does not have to necessarily take physical damage). This could be used for a reaction to an attack such as making a guard intervene.

OnOpen: This script fires when the placeable object gets opened. It implies that the placeable object's HasInventory function is enabled since otherwise there is no way to open the placeable object. This event will generally be used to make something happen when players have touched the placeable object (shut off the lights, lock a door, make a guard hostile, etc.).

OnSpellCastAt: The script attached to this event fires when the object becomes the target of a spell.

OnUnlock: This script fires when the placeable object gets unlocked. Of course, you will first need to lock it in the "Lock" properties of the object.

OnUsed: This script fires whenever a player attempts to use an object. This will happen only if the "Useable" property of the object is enabled in its "Basic" properties and if the "HasInventory" property is disabled. (Otherwise using the placeable object will automatically open the inventory and stop whatever script you have added here). If these properties are set properly, the OnUsed event will fire when a player clicks on the placeable object.

If you want the conversation file placed in the "Advanced" properties of the placeable object to fire when the player clicks on it, place the following script in the OnUsed event:

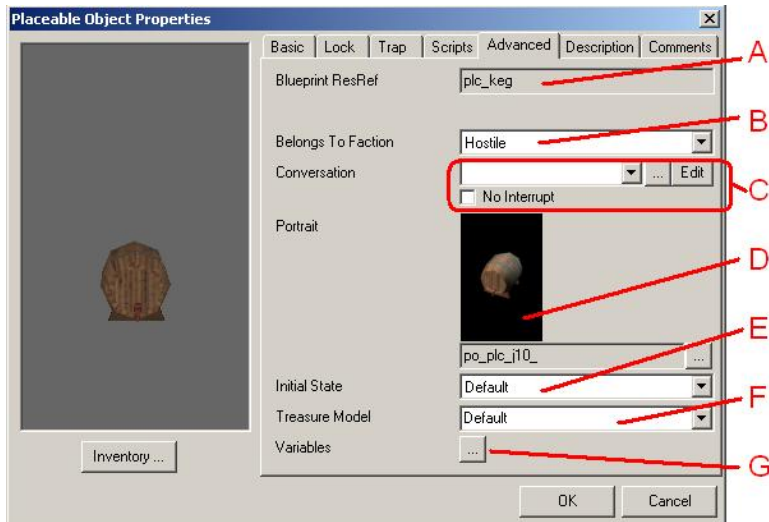
```
void main()
{
    object oObject = OBJECT_SELF;
    object oPC = GetLastUsedBy();
    AssignCommand(oPC, ActionStartConversation(oObject));
}
```

OnUserDefined: The script attached to this event fires whenever an event is signalled to the object. This requires intermediate scripting knowledge. Typically this script is a switch/case statement that does something different depending on the signal (numerical value) sent to it by an event that calls the script.

8.3.5 The Advanced Properties of a Placeable Object

As usual, the advanced properties of the placeable object are those that do not fit in any other categories. The following screen will appear if you click on the “Advanced” tab while the Placeable Object Properties screen is opened:

Image 8.3.5 – Placeable Object Properties – Advanced Tab



- A** Blueprint ResRef: The blueprint ResRef is the unique reference identifying the item blueprint from which this placeable object was created. This information is mainly useful when you wish to create a placeable object through a script, but comes in handy also when you want to update all placeable objects already placed in the module. You do this by making a modification to the blueprint, then right-click on the blueprint and select the “Update Instances” from the dropdown menu.
- B** Belongs To Faction: All default placeable objects are set to the Hostile faction and it should be left that way just to be sure that nothing bad happens when players try to bash it. An exception to this rule might be that you want to implement some scripts in the OnPhysicalAttack event script that would call all allies for help when the placeable object is under attack (or similar scripts). Unless scripts are added, the Faction of a placeable object does not have any other effect.
- C** Conversation: You can add a conversation file to a placeable. Please refer to the OnUsed event (see Sub-Section 8.3.4) for full details on how to make sure that the conversation file will get activated when players click on the placeable object.
- D** Portrait: The portrait of the placeable object can be changed by either clicking on it or on the “...” button.

- E** Initial State: Many placeable objects have animations, like torch brackets that can be either ignited or not, lamps that can be lit or not, etc. You can determine here which one of these animations you want your placeable object to play initially.

If you want players to be able to turn on and off the animation made by a placeable object, make the placeable usable (by enabling the box to that effect in its Basic properties) and add the following script in its OnUsed event (in the Scripts properties):

```
void main()
{
int nState = GetLocalInt(OBJECT_SELF, "state");
if(nState == 0)
{
PlayAnimation(ANIMATION_PLACEABLE_ACTIVATE);
SetLocalInt(OBJECT_SELF,"state",1);
}
else
{
PlayAnimation(ANIMATION_PLACEABLE_DEACTIVATE);
SetLocalInt(OBJECT_SELF,"state",0);
}
}
```

If the placeable object is one that can be opened and closed (like placeable doors and trap doors for example), then you should use the following script instead:


```

void main()
{
object oUser = GetLastUsedBy();
location IWP = GetLocation(GetWaypointByTag("***insert tag of waypoint where the door
leads***"));
int nState = GetLocalInt(OBJECT_SELF, "state");
if(nState == 0)
{
PlayAnimation(ANIMATION_PLACEABLE_OPEN);
SetLocalInt(OBJECT_SELF,"state",1);
}
else
{
AssignCommand(oUser, JumpToLocation(IWP));
PlayAnimation(ANIMATION_PLACEABLE_CLOSE);
SetLocalInt(OBJECT_SELF,"state",0);
}
}

```

- F** Treasure Model: The treasure model is what players will see appearing on the ground if the placeable object is bashed. Of course, for the treasure model to appear the placeable object must have items in its inventory.
- G** Variables: The "Variables" assigned to a placeable object may be used in a script, but otherwise they do not have any effect.

8.3.6 The Description of a Placeable Object

The "Description" tab on the Placeable Object properties screen will show you a large box in which you can write anything you like. All pre-generated placeable object blueprints come with a standard description, which you can modify. The description can be seen by players, if the placeable object is useable, by right-clicking on it and choosing the "examine" option from the radial menu.

8.3.7 The Comments on a Placeable Object

The "Comments" tab on the Placeable Object properties screen will provide you with a large text window in which you can write anything you like (notes for future use of the placeable, comments on the role of the placeable object in your area, or anything else that may be useful to the builder). The comments that you place there have no effect whatsoever on the placeable object.

8.4 Designing Tips

Do not place too many placeable objects on a battleground: Bioware recommends placing no more than two placeable objects per tile in an area. The reason for this is that your engine may have a difficult time finding an appropriate path when player characters, henchmen and NPCs need to move to a specific location. Furthermore, if too many placeable objects are on the path of a NPC, some scripted movements (such as one NPC approaching another to start a conversation or those in a cutscene) will be unreliable and often not occur at all, potentially breaking the story line. You can certainly add more placeables than this recommended limit of two in non-populated areas or in areas where there is no combat (and therefore limited movements are made), but you should really stick to this rule in areas where battles or large combats are supposed to happen to avoid players experiencing lag.

Activate the plot flag on all plot related placeables: When a placeable object needs to be used to complete the module, do not forget to enable its plot flag (in its advanced properties). There will always be players that will try to bash placeable objects, most of the time because they think that this is what is required from them.

Do not paint large placeable objects on the grid lines: Bioware has recommended not to place any placeable on the tile lines (the red lines delimiting each tile). This might cause some serious lag due to pathfinding issues for nearby creatures.

Add descriptions to placeable objects that can be used: A prudent player will examine a usable placeable object before using it. Prudent players will always appreciate the fact that you have added custom description giving them information about what they are looking at.

8.5 How to...

8.5.1 ...make a placeable object engage in a conversation

Please refer to the comments made to that effect in the OnUsed event of the placeable object. (See Sub-Section 8.3.4.)

8.5.2 ...make my placeable drop crafting items when destroyed

Make sure the placeable has the script `x2_o2_dead` or if you are using a custom script for your items have it call `craft_drop_placeable()`; see the example below.

```
#include "x2_inc_compon"
void main()
{
    craft_drop_placeable();
}
```

8.5.3 ...adjust the position of a placeable object

Object Placement

A much under rated tool when decorating an area with placeables is the "Adjust Location" tool. Without it, builders would be limited to their initial mouse click or at best, where they can click and drag an already placed object to. Trying to drag a placed object in the first place is not without problems, particularly in the case where you've placed one object on top of another such as a plate upon a table. Left clicking such objects usually results in them re-positioning themselves on the floor underneath the table. Knowledge of how to use the location adjuster will save you hours of guessing where to click on your supporting placeables. Not only that, but it has other benefits too, such as disabling flicker in game where two objects overlap one another, and making standard objects more unique in the manner in which you place them.

There are two ways in which the tool can be called up from within the toolset. One is to simply right click on the object you want to adjust, then select "Adjust Location" from the resulting pull down menu. The other method is to right click on the object name within your placeable list on the left hand side of the toolset. The latter is recommended, as this method is available on some objects where the former method isn't an option, such as doors (yes, you can re-locate them too!).

Once you select "Adjust Location", you are presented with a window that allows you to adjust four parameters, consisting of X, Y, Z and Bearing. X and Y are self explanatory, being the co-ordinates of the object, while Z refers to the height, with 0.00 being ground level. Bearing refers to the orientation of the object. Height (Z) is probably the most useful of the parameters, allowing you to fine tune overlapping flat objects. As an example, let's suppose you've placed a carpet, then go on to place a sheet of paper upon the carpet. If both objects are at the same height, they will likely flicker in game, or at worst, not even display the paper, particularly if you left clicked on it again within the toolset. You can get round this by adjusting the height of the paper to 0.01, so that it will always be above the carpet. This same method can be used on overlapping objects such as two joining bar areas to disable the flickering that results in game.

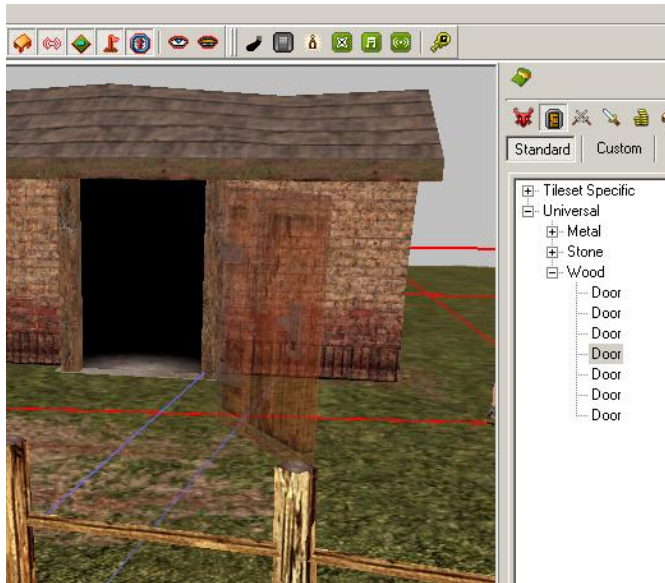
On larger objects, you can put the height into a negative range so that part of the object becomes buried within the ground, allowing you to make shorter bookcases for example. If you are playing around with the height of an object in this manner, you can click "Apply" in the Adjust Location window so that you can see the result of your change and continue to fine tune it until satisfied.

Once you become familiar with the tool, you'll be kicking yourself for not having discovered it earlier! Use it to place paintings and windows to a more visually pleasing height, or stack kegs on top of one another against a wall. Be creative, and suddenly your areas will have a more unique feel to them with surprisingly little effort.

CHAPTER 9 Doors

When you are building your areas, you will notice that some tiles that you have placed already come with doors (like cave exits, temples, etc.) while others come with big black holes in which a door still needs to be placed (like most buildings). For those tiles where doorways are left empty, you will need to add a door yourself from the Doors Palette a show in the image below.

Image 8 – Adding a door



The “Tileset Specific” doors in the palette are those that come automatically with buildings/tiles contained in the tileset you chose when you created your area. There is absolutely no reason for you to choose a door in the “Tileset Specific” division of the doors palette since these doors do not fit in doorways other than those for which they were designed. Your choices are therefore limited to the doors that are listed in the “Universal” category of door blueprints.

When you want to place a door in a doorway, select one from those that are listed in the “Universal” category of door blueprints. Your cursor will be changed into a spinning door. When your cursor is over an empty doorway, the door will place itself properly. Click to leave it there.

9.1 The Properties of a Door

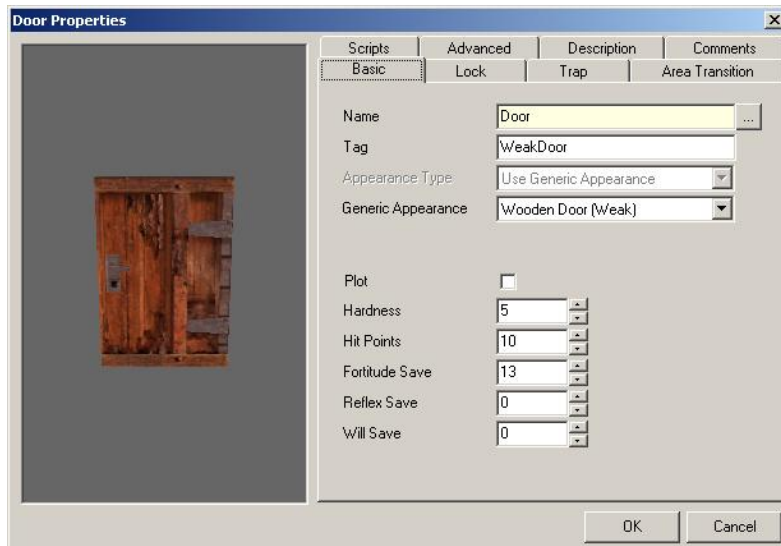
To access the properties of a door, right-click on it and choose the “Properties” option in the dropdown menu that appears. The door properties screen that then appears is divided into 8 categories of properties, each of them covered by a sub-section below.

You will notice that doors and placeable objects are a lot alike in many respects. Do not feel confused if you have a “d  j   vu” feeling when you read this section of the manual.

9.1.1 The Basic Properties of a Door

The basic properties of a door look a lot like those of a placeable object and they serve the same purpose.

Image 9.1.1 – Door Properties Screen – Basic Tab



Name: The name of the door is for you to choose. You might however want to leave the name “Door” rather than giving it a code name that permits you to know where this door leads. The main reason for this is that when players try to bash a door, the system shows the name of the door in the message screen to show the amount of damage done by each successful hit made against the door. For players, receiving a message like: “d_temtar382: Successful hit, 3 points of damage” does not make much sense.

Tag: The tag of the door is used not only to identify it in a script, but also to link it with another door or an area transition. It is therefore important to give a unique tag to each of the doors that can be used in your module to ensure that transitions between them work properly.

Appearance Type / Generic Appearance: The available appearances for doors are quite limited. Each appearance type, except for the “Force Field” appearance type, has its own door blueprint in the palette. Changing the appearance of your door will only be esthetical, it will not change the other properties of the door to match those of the door blueprint that uses the appearance type you have selected (i.e. stone doors in the palette are harder than wooden doors). If however you change the appearance of a wooden door to a stone one, the resulting door will remain weaker and only its appearance will change.

Plot: If you tick the “Plot” box, the door cannot be destroyed (i.e. bashed). Do not forget that doors can not only be destroyed by players attacking/bashing them directly, but also inadvertently by area of effect spells.

Hardness: The hardness value tells you how many damage points the door can absorb without being affected by a successful hit. As an example, if the door has a hardness value of 5, a player will need a weapon that does at least 1d6 damage to be able to bash it (in which case 1 point of damage will be done on a roll of 6, since the other 5 damage points will be absorbed). Any weaker weapon will be ineffective.

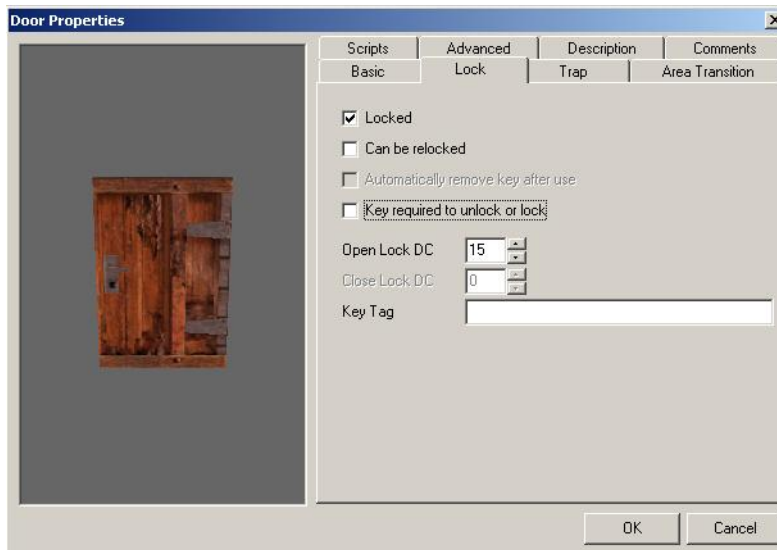
Hit Points: This value is the same as for any creature or player character. It determines how many damage points you have to inflict on the door before a player destroys it.

Fortitude/Reflex/Will Save: A door can be the target of a spell. This property determines how well the door resists spells that allow for a fortitude/reflex/will save.

9.1.2 The Properties of the Door Lock

What would be the use of a good rogue if doors were never locked? On the other hand, do not forget that not all parties of players will have a good rogue amongst them when you are editing the properties of the door lock.

Image 9.1.2 – Door Properties Screen – Lock Tab



Locked: Ticking this box will confirm that the door is locked and therefore players cannot open it without first unlocking it or bashing it.

Can be relocked: Tick this box to allow players to relock the door (which makes more sense in permanent world modules than in solo player modules). Note that it is always possible to relock a door through a script, even if this box is not ticked.

Automatically remove key after use: If you have ticked the “Key required to unlock or lock” option, then you might want to activate this option as well if (i) players only have to unlock the door once; and (ii) the key that is required has no other use than to open this specific door. Although it does not make much sense to have a key vanish once the player uses it, it does however help them to keep their inventory clear of items/keys that are no longer relevant.

Key required to unlock or lock: By ticking this box, it will no longer be possible for players to unlock the door and therefore to open it unless they first find the required key. You will need to place the tag of the item that will permit players to open the door in the “Key Tag” parameter.

Keep in mind however that ticking this box will not prevent players from bashing the door. If you want to make it invulnerable to bashing, tick the “Plot” box appearing in the Basic properties of the object.

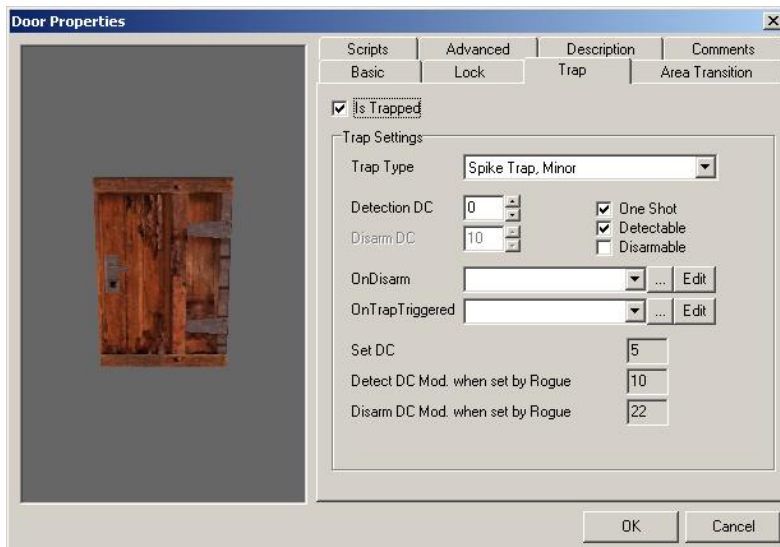
Open/Close Lock DC: These parameters can be adjusted when a key is not required to unlock the door. They determine the level, on a d20 die roll, required to successfully unlock or lock it.

Key Tag: Here you must add the tag of the item that players must have in their inventory to be allowed to unlock the door when the “Key required to unlock or lock” option is activated. The item does not have to be a key: it can be a sword, a wand or any other item type. Keep in mind that if the “Automatically remove key after use” option is activated, the item identified here will vanish when the door is first opened, which might be a dirty trick to do to players if the key is a sword.

9.1.3 The Trap on a Door

A door can of course be trapped, and all the parameters to that effect are located under the “Trap” tab of the door properties screen as shown below.

Image 9.1.3 – Door Properties Screen – Trap Tab

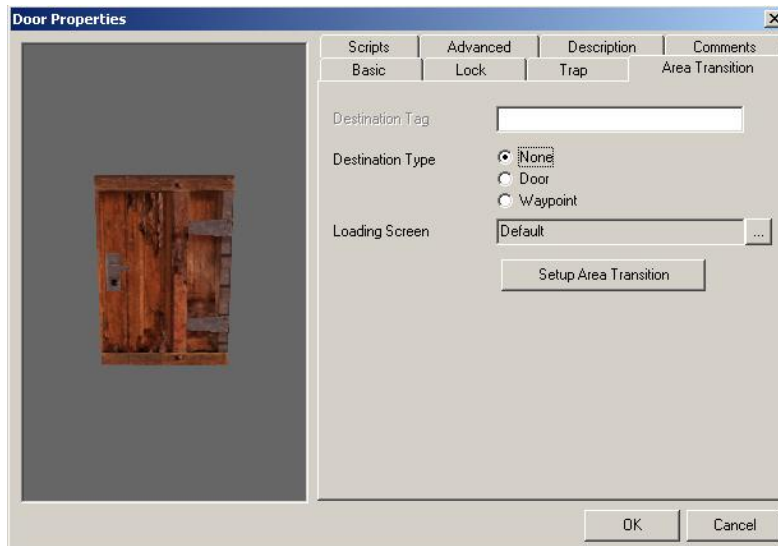


These settings are exactly the same as those available for traps that are placed on a placeable object. We therefore refer you to Sub-Section 8.3.3 for more comments on these parameters.

9.1.4 The Area Transition Parameters of a Door

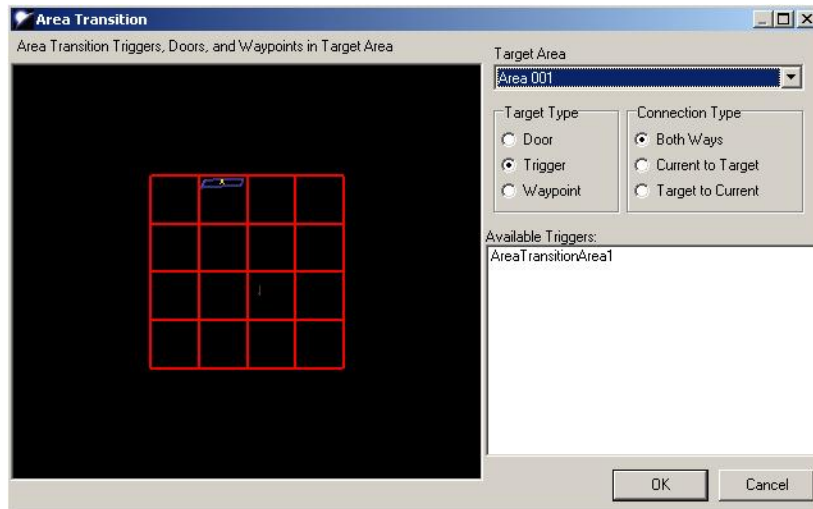
Doors are used to link areas together. In order to implement this, you must first link your door to its destination, which can be found in the "Area Transition" parameters of the door.

Image 9.1.4a – Door Properties Screen – Area Transition Tab



Linking a door to its destination is in fact quite easy. You do not have to fill out the information required in the screen above by yourself, the toolset provides you with an easy to use tool for that purpose. Click on the "Setup Area Transition" button and the following screen will appear:

Image 9.1.4b – Area Transition Screen



IMPORTANT NOTE: You must first assign a **unique tag** to your door and to the destination (whether it is a trigger, a door or a waypoint) for this tool and your area transition to function properly.

In order to set up the transition, you will need to follow this process:

Step 1: Choose the area your door leads to from the “Target Area” dropdown menu. The grid of the selected area will then appear on the black portion of this screen.

Step 2: Determine what type of target the destination of your door is. It can be another door, a trigger or a waypoint. Depending on which type you have selected, the tags of the available objects in the area you have selected in step 1 will appear in the “Available Doors/Triggers/Waypoints” list.

Step 3: Decide if the destination of your door leads back to your door when used the other way around, and make your choice in the “Connection Type” parameter. If you configure your transition to work “Both Ways”, the area transition properties will be set automatically on both your door and its destination, without any need to adjust the properties of the destination further.

Step 4: Choose the destination from the ones that are available in the “Available Doors/Triggers/Waypoints” list. If you are unsure of the tag of the destination target of your door, keep in mind that selecting a destination in the list will highlight its location in the grid appearing in the black screen, which can be useful to get a better idea where these possible targets are located in the destination area.

Once you have gone through these 4 steps click on the “OK” button and all the area parameters will be used to adjust the “Destination Tag” and “Target Tag” properties of your door accordingly.

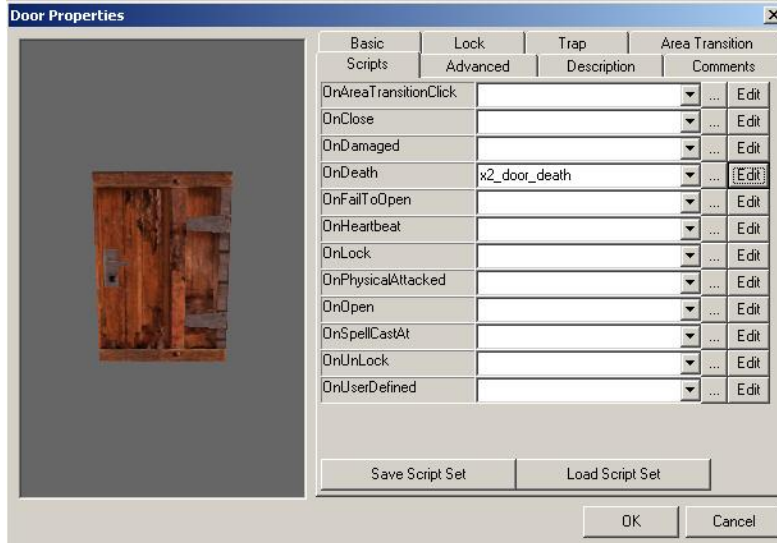
The only other property appearing under the “Area Transition” properties of the door is the “Loading Screen” properties. As seen in Sub-Section 3.2.5 of the manual, it is possible to assign a specific loading screen to each door that players will see when the target area is being loaded. This door

property permits you to designate a different loading screen that players will only see if they enter the destination area through this specific door.

9.1.5 The Script Events of a Door

As for any other object, each door comes with its load of script events that can be used to make your door exactly as you want it to be.

Image 9.1.5 – Door Properties Screen – Scripts Tab



OnAreaTransitionClick: This script fires when a player clicks on the area transition found behind the door (the light blue thing that appears when you place your mouse in a doorway). If the area transition is set on the door, the player will automatically be transported to the destination without any need to add a script here.

OnClose: This script fires when the door gets closed, by whatever means.

OnDamaged: The script attached to this event fires when the door is damaged.

OnDeath: The script attached to this event fires when the door is destroyed for any reason (whether because it is bashed, affected by a spell, etc.). DestroyObject() will not trigger this event, although DM kill will (with an invalid last killer).

Scripts that execute in this event should have their actions take effect immediately. Using DelayCommand() will not work. Although AssignCommand() could be used to place an instruction in the action queue of another object, for example an invisible object placeable or the module itself. Also note that if DestroyObject(OBJECT_SELF) is used in the OnDeath event, it will also cause associated UserDefined events, ie the Death event, to not execute at all.

The default script that appears in the OnDeath event of most door objects is a script that will create wood that can be crafted if the craft item scripting option that comes with the HotU expansion is enabled on the module.

OnFailToOpen: This script will fire when the door is locked and a player attempts to open it.

OnHeartbeat: The script associated with this event fires every six seconds. This is good for a script that needs to check for something repeatedly or perform an action regularly. Be aware that too many OnHeartbeat scripts can be processor costly. Use other types of event scripts when possible.

OnLock: This script fires when the door gets locked. You will need to enable that function in the "Lock" properties of the object to allow it.

OnPhysicalAttacked: The script attached to this event fires each time the door takes physical damage. This could be used as a reaction to the attack, such as making a guard intervene.

OnOpen: This script fires when the door gets opened.

OnSpellCastAt: The script attached to this event fires when the door becomes the target of a spell.

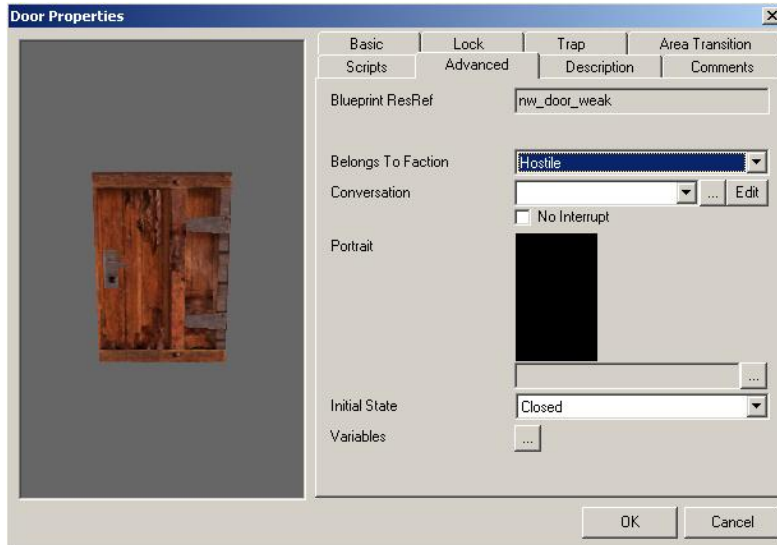
OnUnlock: This script fires when the door gets unlocked. You will need to lock it in the "Lock" properties of the object to allow it.

OnUserDefined: The script attached to this event fires whenever an event is signalled to the object. This requires intermediate scripting knowledge. Typically this script is a switch/case statement that does something different depending on the signal (numerical value) sent to it by an event that calls the script.

9.1.6 The Advanced Properties of a Door

A door object has the same advanced properties as a placeable object, as seen in the image below.

Image 9.1.6 – Door Properties Screen – Advanced Properties Tab



Blueprint ResRef: The blueprint ResRef is the unique reference identifying the door blueprint from which this door was created. This reference is mainly useful when you want to update all doors already placed in the module in accordance with a modification you have made to the blueprint that was used to create them (this can be done by right-clicking on the blueprint and selecting the "Update Instances" in the dropdown menu).

Belongs To Faction: All doors are set on default to the Hostile faction and it should be left that way just to be sure that nothing bad happens when players try to bash it, An exception to this rule might be that you want to implement some scripts in the OnPhysicalAttack event script that would adjust the reputation of the player that is bashing it (or similar scripts). Unless scripts are added to the door, its faction does not have any effect.

Conversation: You can add a conversation file to a door. This is quite useful when you want a conversation to be triggered when a player attempts to open a locked door, to make it as if someone on the other side of the door is enquiring what the player wants. The conversation file will not launch unless you add a script to that effect in the appropriate event of the Scripts properties of the door. In the previous example, the following script would need to be added in the OnFailToOpen event of the door:

```
void main()
{
object oPC = GetClickingObject();
object oDoor = OBJECT_SELF;
AssignCommand(oPC, ActionStartConversation(oDoor));
}
```

Portrait: The portrait of the placeable object can be changed by either clicking on it or on the “...” button.

Initial State: You can change the initial state of the door here, making it so that the door is already opened. All doors are initially closed by default.

Variables: The “Variables” assigned to a door may be used in a script, but otherwise (with one exception) they do not have any effect. The one variable that may have an effect on doors that have already been coded in the game is:

X2_FLAG_DOOR_RESIST_KNOCK

If you set this variable as an integer on a door and you give it the value of 1, then the spell “Knock” will no longer work on this door and feedback will be provided to the players to that effect. If you give it a value of 2, not only will this spell not work but no feedback at all will be provided to players to the effect that the spell has failed.

9.1.7 The Description of a Door

The “Description” tab on the door properties screen will show you a large box in which you can write anything you like. All pre-generated door blueprints come with a standard description which you can modify. The description can be seen by players when they right-click on the door and choose the “examine” option from the radial menu.

9.1.8 The Comments on a Door

The “Comments” tab on a door will provide you with a large text window in which you can write anything you like (notes on the use of the door or anything else that can be useful to the builder). The comments that you place there have no effect whatsoever on the door and cannot be seen by players.

9.2 Designing Tips

Do not leave empty doorways or doors that lead to nowhere: Most buildings will have doorways and some tileset terrain (like the city exterior tileset) also creates buildings with doorways. Most doorways will be empty and builders often wonder what to do with these doorways when the buildings are not used in the module. Two solutions to this problem are generally adopted by builders:

- (1) Add doors in these unused doorways, lock the doors, make them a plot item and activate the “require key to lock/unlock” option so that they cannot be opened or bashed by players. The doors then becomes part of the decor and has no use at all.
- (2) Add doors in these unused doorways and, in the “Area Transition” properties of the door, choose a door or waypoint in the “Destination Type” property without however properly putting in place an area transition. By doing this, the usual light blue area transition will appear behind the door when it is opened by players but nothing will happen when they click on the area transition. Then, to explain why nothing happens when players click on the area transition, the following script is added in the OnAreaTransitionClick event of the door:

```
void main()
{
    SendMessageToPC(GetClickingObject(), "There is nothing of interest for you inside this
    building.");
}
```

It should not take half an hour to bash a door: When adjusting the “Hardness” and the “Hit Points” of a door, do not forget that players should not necessarily have time to make their breakfast and wash themselves before they succeed in bashing the door (unless of course you want to dissuade players from bashing doors, but then you should make the door a “plot” object rather than giving it 5,000,000 hit points). Do not forget also that players playing wizards usually do not have a very damaging weapon and they are reluctant to empty their spell book on a door. The bottom line is stay reasonable in adjusting the basic properties of a door and allow players to bash it reasonably easily if it is meant to be bashed.

9.3 How to...

9.3.1 ...make my doors close themselves after a certain delay

This can be done by adding this very simple script to the OnOpen event of the door:

```
void main()
{
    float fDelay = 20.0;
    DelayCommand(fDelay, ActionCloseDoor(OBJECT_SELF));
}
```

The delay can of course be adjusted at will. If you have put an area transition on your door that leads to another door, door, you will have to add this script to both doors or it will not work.

Axe Murderer:

I found that if you assign the DelayCommand to the area instead of the door...

You can use this instead:

Delayed opening and/or closing of doors will be more responsive if you make the area responsible for maintaining the delay instead of having the door do it. Like so:

```
AssignCommand( GetArea( oDoor), DelayCommand( 6.0, AssignCommand( oDoor, ActionCloseDoor( oDoor))));
```

Another thing that works better is to make a placeable object in the area perform the action instead of to the door itself, if a placeable is available in the area, because placeables have a higher priority than doors do:

```
NWScript:  
object oPlaceable = GetNearestObject( OBJECT_TYPE_PLACEABLE, oDoor); if(  
GetIsObjectValid( oPlaceable)) AssignCommand( oPlaceable, ActionCloseDoor( oDoor));  
else AssignCommand( oDoor, ActionCloseDoor( oDoor));
```

9.3.2 ...Make my doors drop crafting items when destroyed

Make sure the OnDeath event for your item has the x2_door_death script assigned to it or if you are using custom scripts make sure it calls craft_drop_placeable(); see the example below.

```
#include "x2_inc_compon"  
void main()  
{  
    craft_drop_placeable();  
}
```

9.3.3 ...make free standing doors

Paint a tile with a door, move the door out of the way by right clicking on it in the Module Structure pane on the right. Delete the tile with the doorway in the Main View Window and you've still got a door that you can move wherever you want.

CHAPTER 10 Encounters

If you are building a module that is intended to be played in multiplayer mode or by a wide range of levels for player characters, then you might want to design most, if not all, your combat scenes with encounter objects.

Basically, an encounter is a trigger painted on the ground in an area. When a player character walks on this trigger, the encounter object determines the strength of the party of players and spawns enough creatures from a designated group to generate a combat that is scaled to meet your challenge criteria. Does that sound complicated? You will find that it is not.

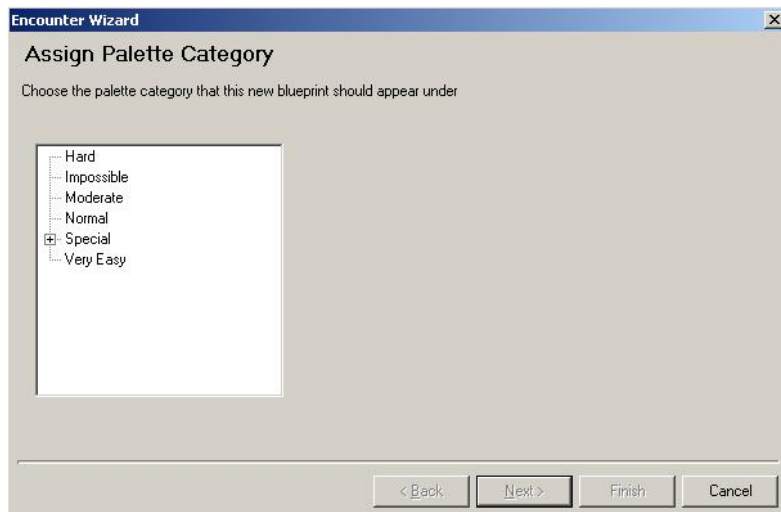
As for any other object, encounters are added to your module from a blueprint. The “Standard” blueprints are categorized depending on the difficulty level of the group of creatures that can be spawned when the encounter is triggered.

To add an encounter to your module, select the appropriate blueprint in your palette. Your cursor will then turn into a paintbrush. Draw the zone where you want the encounter to be triggered when walked upon by players (double-click when the polygon that you have drawn is satisfactory) and... that's it. When players walk into the zone that you have drawn, the engine will generate an encounter that meets the criteria of your encounter object. The present chapter is aimed at guiding you in creating and editing your own encounter blueprints.

10.1 Creating an Encounter Blueprint with the Encounter Wizard

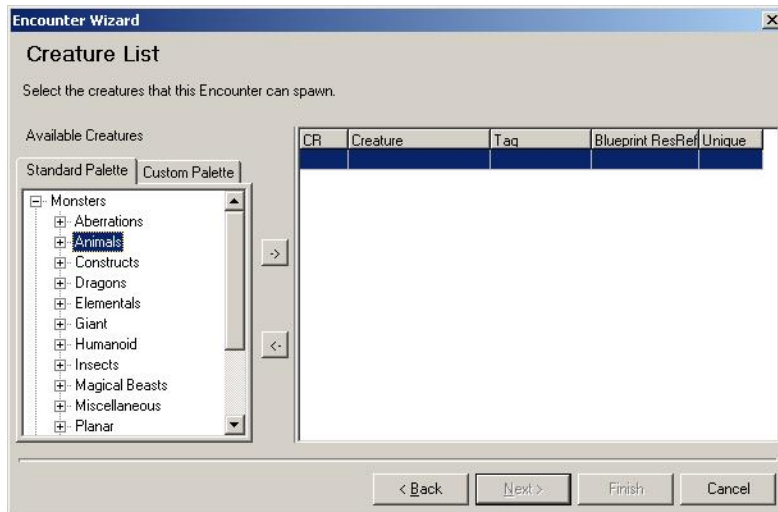
Using the Encounter Wizard is only the first step in creating your own blueprints. You will find that this Wizard helps you to generate a new blueprint but does not provide any help in adjusting this blueprint.

Image 10.1a – Encounter Wizard – Step 1



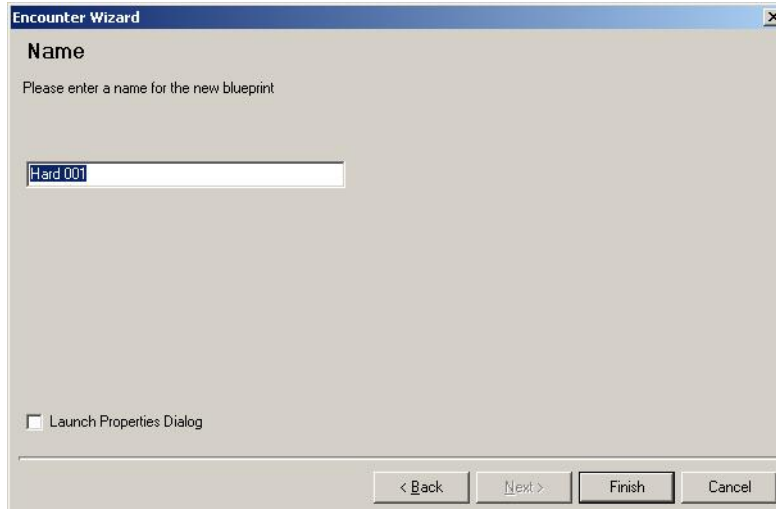
STEP 1: Determine in which palette category your new blueprint will appear. This is only a question of archiving the encounter, the category that you choose will in no way affect the encounter that you will be creating. It will NOT adjust the difficulty level of your encounter and it will NOT limit the available creatures list to scale with the category you have chosen.

Image 10.1b – Encounter Wizard – Step 2



STEP 2: Determine which creatures can be spawned when players trigger the encounter. Not all creatures in this list will be spawned, the engine will choose creatures randomly from the list you will be making, selecting the ones that fit with the parameters of your encounter in consideration of the overall strength of the party of players. This list of creatures can be edited later on, so do not bother to make the list perfect at this stage.

Image 10.1c – Encounter Wizard – Step 3



STEP 3: And now it is already time for you to name your encounter, the wizard has done everything he could for you. Next, you will have to edit the properties of your encounter blueprint to adjust its parameters adequately. Or you can select the “Launch Properties Dialog” box in the encounter wizard, prior to closing the wizard, if you want to edit the properties of your encounter blueprint.

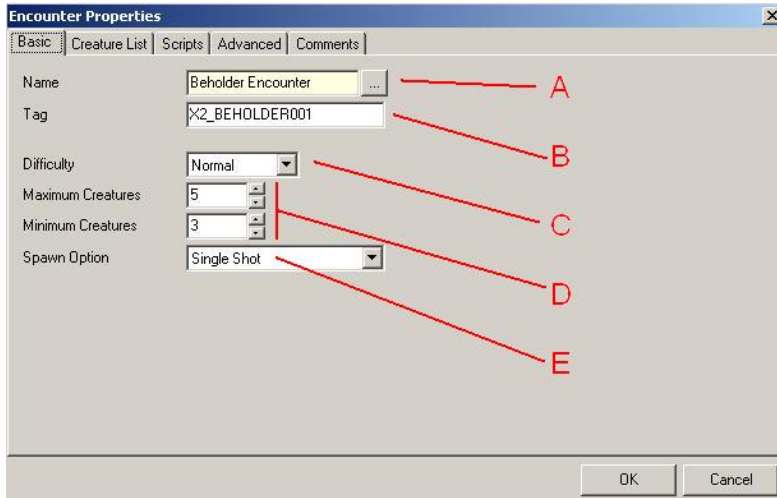
10.2 The Properties of an Encounter

The Encounter Properties Screen can be accessed by right-clicking on the encounter blueprint you wish to edit and choosing the “Properties” option from the dropdown menu that appears.

10.2.1 The Basic Properties of an Encounter

It is very important that you edit the basic properties of your encounter first. Although there are not many of them, these basic properties will determine what parameters will be available under the other property tabs.

Image 10.2.1 – Encounter Properties Screen – Basic Tab



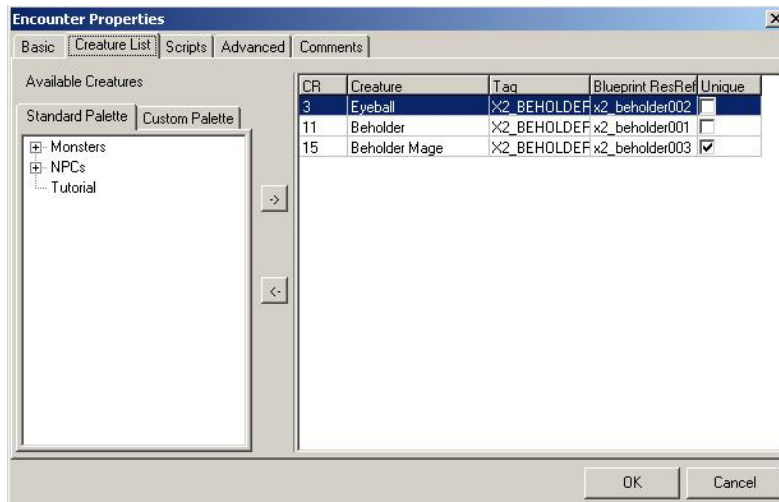
- A** Name: The name of the encounter object is only for your reference. This name will never appear to players and will therefore not have any impact on the gameplay.
- B** Tag: As for any other objects, the tag of an encounter is primarily used to identify this encounter in a script (this is mainly useful when you want to change an inactive encounter to its active state or vice versa, see Sub-Section 10.2.4 for more details on this topic). If it is not your intention to have scripts applied to your encounter, then you may leave the default tag as is.
- C** Difficulty: This property permits you to determine if the encounter is intended to be very easy, easy, normal, hard or impossible for players. This parameter is taken into consideration by the engine when the encounter is triggered in order to determine the quantity and the challenge rating of the creatures that will be spawned. Note however that this parameter will NOT adjust the creatures in the creature list. Therefore, if the encounter is triggered by a 40th level barbarian and only goblins are available in the creature list, the encounter will spawn weak goblins even though you have set this encounter to be “impossible”. The reverse is also true. The encounter object will work within the parameters that you give to it, not beyond that.
- D** Minimum/Maximum Creatures: You can set a minimum of 1 and a maximum of 8 creatures that will be spawned by the encounter object. This is one of the parameters that will be respected at all times by the encounter when it is triggered, no matter what the other parameters are. Therefore, if you set the minimum at 6, then at least 6 creatures will be spawned even if the encounter is triggered by the weakest bard in the Forgotten Realms. The more flexible these parameters are, the better the encounter will scale to the strength of the player that triggers it.
- E** Spawn Option: Here you can determine if the encounter can be spawned only once (Single Shot) or if it can be spawned on more than one occasion (Continuous). If you set this

parameter to “Continuous”, you will be able to set the conditions under which the encounter will respawn under the “Advanced” properties tab.

10.2.2 The Creature List of an Encounter

When the encounter is triggered, the engine spawns creatures only from those that are listed in this screen, taking into account the other parameters you have set for your encounter.

Image 10.2.2 – Encounter Properties Screen – Creature List Tab



To add creatures to this list, find the appropriate creature blueprint in the palette window and click on the arrow button.

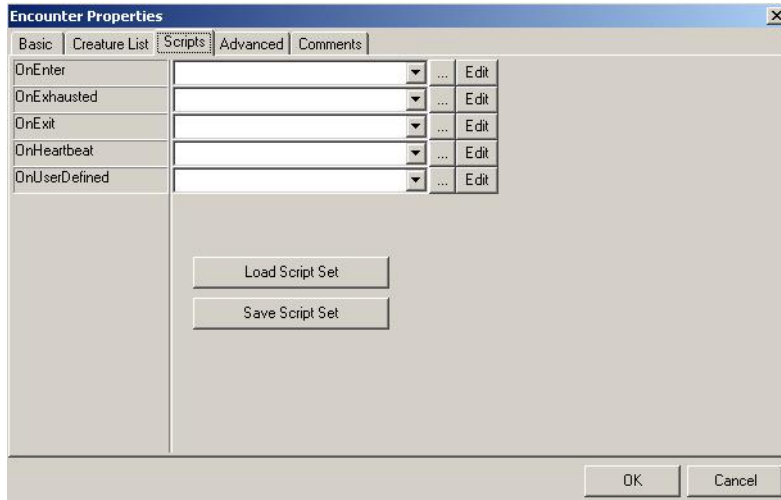
Pay a special attention to the “CR” column of your creature list. (“CR” stands for “Challenge Rating”.) This value is taken into consideration by the engine when it generates and scales an encounter that will meet the set difficulty of the encounter, in addition to the strength of the party of players and the other parameters that you have set for your encounter object. Therefore, if your module is designed to be playable by a wide level range of characters, it is important to adjust your creature list so that creatures within a wide range of “Challenge Rating” are available for the engine to pick from. Otherwise your encounter will probably not scale adequately to the party of players that trigger it.

The “Unique” column permits you to identify creatures that can be spawned only once by the engine for this specific encounter object. Keep in mind however that even though a creature is set as “Unique”, this creature may not spawn at all.

10.2.3 The Scripts related to an Encounter

The script events of an encounter object are a lot like those of a trigger object. Since the encounter object is used for only one purpose (which is to spawn creatures), very rarely will you need to add scripts to it.

Image 10.2.3 – Encounter Properties Screen – Scripts Tab



OnEnter: The script attached to this event fires when a creature (and therefore not necessarily a player character) enters the painted boundaries of the encounter. The spawning of creatures by the encounter object will work without any need to add a script here, so this script event should be used to add events in addition to the spawning of creatures. For example the setting of variables on the PC or the spawning of other objects or the triggering of other things.

NOTE: It is not possible to prevent the firing of the encounter with scripting on this event. To control the firing of the encounter a second trigger placed on top of and encompassing the encounter trigger must be used.

OnExhausted: The script attached to this event fires when the last member of a party of creatures spawned by an encounter dies. It is important to note however that this event does not work properly and should not be relied upon. In fact, when a player dies and reloads after an encounter has spawned, but before killing all of the creatures generated by the encounter, the OnExhausted event for that encounter will not fire when the creatures are thereafter killed.

OnExit: The script attached to this event fires when a creature (and therefore not necessarily a player character) exits the painted boundaries of the encounter.

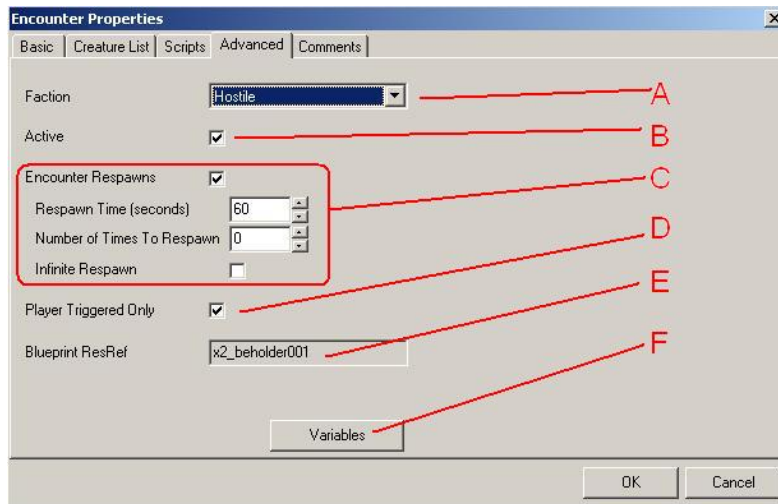
OnHeartbeat: The script associated with this event fires every six seconds. This is good for a script that needs to check for something repeatedly or perform an action regularly. Be aware that too many OnHeartbeat scripts can be processor costly. Use other types of event scripts when possible.

OnUserDefined: The script attached to this event fires whenever an event is signalled to the object. This requires intermediate scripting knowledge. Typically this script is a switch/case statement that does something different depending on the signal (numerical value) sent to it by an event that calls the script.

10.2.4 The Advanced Properties of an Encounter

The Advanced Properties provide you with more parameters and options to be set on the encounter object.

Image 10.2.4 – Encounter Properties Screen – Advanced Tab



- A** **Faction:** The faction of the encounter object is taken into consideration when determining if the encounter is fired or not when a player enters within its boundaries. An encounter will fire only if the player triggering has a hostile reputation towards the faction set for the encounter. As an example, if you set the faction of your encounter to “Commoner”, then a player will trigger the encounter only if he has previously done bad actions towards members of the “Commoner” faction that has made its reputation shift to hostile.

It is a common mistake to believe that this “faction” property sets the faction of the creatures that will be spawned through this encounter, which is not the case. All creatures spawned by the encounter will keep the faction that was attributed to them on their blueprint.

Defender/Merchant or Commoner faction encounters can be used to spawn in Defenders if a Hostile faction creature wanders too close to town and you wish to make sure that the NPCs in that town don't get destroyed by wandering monsters. PCs that are neutral or friendly to Defender faction will not trigger these encounters, only Hostile NPCs will.

- B** **Active:** Ticking this box confirms that the encounter is active and ready to be triggered. You may however want to paint encounters that will become active only at a certain moment. An example of this would be to paint encounters with guards spawning, which becomes active only after players have triggered an alarm on a safe in the house of a rich nobleman. To turn an inactive encounter to its active state, you need to call a script on it that would look something like this:

```
void main()
{
object oEncounter = GetObjectByTag("ADD_ENCOUNTER_TAG_HERE");
SetEncounterActive(TRUE, oEncounter);
}
```

- C** Encounter Respawn Conditions: These properties become available only if you have set the encounter as "Continuous" under its basic properties. The "Respawn Time" will start to run only from the moment that all creatures previously spawned by the encounter are killed (or otherwise destroyed by a script).
- D** Player Trigger Only: If this property is not enabled, than any creature will trigger the encounter object. This can be very nasty in a populated area where NPCs walk randomly.
- E** Blueprint ResRef: The blueprint ResRef is the unique reference identifying the encounter blueprint from which this encounter object was created. This reference is mainly useful when you want to update all encounters already placed in the module in accordance with a modification you have made to the blueprint that was used to create them (this can be done by right-clicking on the blueprint and selecting the "Update Instances" in the dropdown menu)
- F** Variables: The "Variables" assigned to an encounter may be used in a script, but otherwise they do not have any effect.

10.2.5 The Comments on an Encounter

The "Comments" tab on an encounter will provide you with a large text window in which you can write anything you like (notes about the purpose of the encounter or anything else that can be useful to the builder). The comments that you place there have no effect whatsoever on the encounter and cannot be seen by players.

10.3 Spawning Points

Everyone has played a module in which creatures spawned through an encounter appear from thin air directly ahead of our player character. This is usually because the builder has not taken the time to set spawning points for his encounters.

When an encounter that has no spawning points is triggered, then the engine will set the spawning point as the farthest point, within the boundaries of the painted encounter, from the player that has triggered it. Of course, if the painted encounter is relatively small, the this farthest point will often be right in front of the player that has activated it.

To avoid this, you can manually set one or more spawning points for your encounter that will identify where you want the creatures to be spawned when the encounter is triggered. There are two ways to create a spawning point:

- i) By right-clicking on the encounter you have painted in an area and selecting the "Add Spawn Point" option in the dropdown menu that appears. A point with big arrows on

top of it will then appear on the encounter. This point can then be dragged and dropped wherever you like in your area.

- ii) By selecting the encounter you have painted in an area for which you wish to create a spawning point. Then, while the encounter object is still selected, right-click on the location in your area where you wish to add a spawning point and select the "Add Spawn Point" option in the dropdown menu that appears. A point with big arrows on top of it will then appear, identifying the spawning point at the selected location.

If you add more than one spawn point to an encounter object, the game engine will choose a spawn point that is farthest from the PC that triggered the encounter and not in line of site, if all spawn points are equally distant or all are within line of site of the PC then one is selected randomly.

For your spawning points to be effective, you should consider placing them out of sight from the location of the encounter. You probably understand that placing a spawning point right in the middle of the painted encounter defeats the purpose of these points.

10.4 Designing Tips

Do not place too many encounters in the same spot: You have probably already noticed that encounters are demanding on the game engine. Spawning many objects at the same time often results in a short lag when the encounter is triggered. Therefore, you may want to combine placed creatures and encounter objects to reduce the lag in areas designed to support a lot of battles.

Add spawning points to your encounters in an out of sight location: It is a good practice to have encounters triggering far from the actual place where you actually want the battle to happen (but of course to add its spawning point(s) at the actual battle scene). Otherwise, the creatures that will be spawned when the encounter is triggered will generally appear right in front of the player that triggers it, which is considered a major design flaw (unless it makes sense for the creature to appear from nowhere, like spiders coming down from trees).

Be careful using custom creatures: In general encounters seem utterly unreliable when custom creatures are concerned (especially at higher levels). The Aurora engine seems to get it quite wrong. Whether this is a case of a custom creature list being inherently impossible to balance for the engine's encounter parameters or not is unknown. Certainly the fact that CR seems to play a large part in this process is problematical.

In particular there's a glitch with the unique flag. When minimum and maximum numbers in an encounter design are identical, for example, and all creatures are flagged unique - which is what a modder might do if she were aiming for specific combat design in a SP mod - then the encountered group does not match the encounter design. Instead of a number of unique creatures you end up with duplicates of some and no-shows of others. When minimum and maximum numbers are different flagging a creature(s) as unique almost guarantees that the creature will not appear at all.

If both standard and custom creatures are in a list together then the standard creatures behave as they should. They obey the unique flag in all situations.

It's recommended with custom creatures, especially at higher levels, that the modder makes use of either an OnEnter script in a generic trigger to spawn exactly which creatures she wants to appear, and where, and/or else utilise a system of 'bunched encounters' ie a number of encounters drawn

close together, each with a single custom creature in its list. This seems to be the only sure way of designing combat or other encounters with any specificity.

The good news regarding encounters and custom creatures, however, is that the system works perfectly (ie as for standard creatures) if custom creatures are autolevelled OnSpawn.

CHAPTER 11 Sounds

Sounds will generally make a huge difference, good or bad, in the designing of a module. The generic sounds that come with an area are usually imperceptible and you will therefore need to add placeable sound objects to your module to make it feel more alive.

Like any other object type, sound objects have their own palette of blueprints. Many standard and ready to use blueprints are already available in the palette.

When you place sound objects in your areas, you will notice that most of them fill the screen with a collection of concentric circles. These spheres are there to show you the area of effect of the sound object (i.e. where they can be heard by players).



These spheres make it quite hard for you to see the rest of the area. Therefore, when you are not working on adding sounds to your module, you might want to hide all the already placed sound objects to have a better view of your areas. This can be done by disabling the "Show Sounds" button in the upper tool bar of the toolset editor.

11.1 Creating a Sound Blueprint with the Sound Wizard

It is always easier to work with the already provided sound blueprint in the "Standard" palette and to edit the specific sound objects that you will place in your areas according to your tastes. One of the reasons for this is that it can be very hard to find the sounds that you are looking for as the number of individual sounds available is very large. Another reason is that adding sounds to a sound object may be quite long and tedious (on some machines the toolset may slow down when creating custom sound objects) and it is preferable to start with an object that already covers most of the sounds you are looking for and then modify it to suit your needs.

If you still want to use the Sound Wizard and create your own sound blueprints from scratch, then the following steps will lead you to the creation of such blueprints.

Image 11.1a – Sound Wizard – Step 1



Step 1: Although you have not yet selected a single sound for your blueprint, you are already asked in which category this blueprint will be placed in the Custom palette.

Choosing such a category is for organizational purposes only, your choice will in no way affect the sounds that will be available later on (i.e. even if you choose the “Animals” category, your sound may only be bells ringing if you like).

Image 11.1b – Sound Wizard – Step 2



Step 2: The comments provided by the Wizard on this second step relate to timing are self-explanatory.

Note however that this property may be changed later on (since you have still not even selected a sound yet, you may change your mind) and a third possibility, which is to play the sound only once, is available when you edit the properties of a sound object.

Image 11.1c – Sound Wizard – Step 3



Step 3: Again, the comments provided by the Wizard are self-explanatory.

If you choose one of the “Random Positional” or “Positional” options, editing the properties of your sound blueprint later on will permit you to adjust the position that has been attributed by default to your sound blueprint.

Image 11.1d – Sound Wizard – Step 4



Step 4: Finally, it is time for you to choose the list of sounds that will be used your sound object. Why a list of sounds? Because as seen above during step 2, a sound object is not necessarily a looping sound. It may be a sound that is played only every 20 seconds. And instead of having the same sound played over and over again every 20 seconds, a sound object permits you to establish a list of sounds to pick from when it is time to play it. This allows you to add more sound variety without necessarily having to add many sound objects in your module’s areas.

Note however that if you have chosen your sound object to play “Seamlessly Looping” at step 2 of the creation process, then only one sound can be added to the list (which will be played in a never ending loop).

When you will click on the “Add Sounds...” button, you will have to be patient since many sounds are available and the engine will need some time before it opens a window with all the available sounds in it. Adding Sounds is a topic fully covered in Sub-Section 11.2.1 of this manual and we refer you to this section for more details on this subject.

Image 11.1e – Sound Wizard – Step 5



Step 5: Once you have added at least one sound to your list, you can go to the next and last step, which is to name your new blueprint.

This name is for reference purposes only since players will never see the name of a sound that is being played in the game.

You can select the “Launch Properties Dialog” box in the sound wizard, prior to closing the wizard, if you want to edit the properties of your sound blueprint.

NOTE: The name must be unique as it is used by the wizard to generate the ResRef of the new sound object.

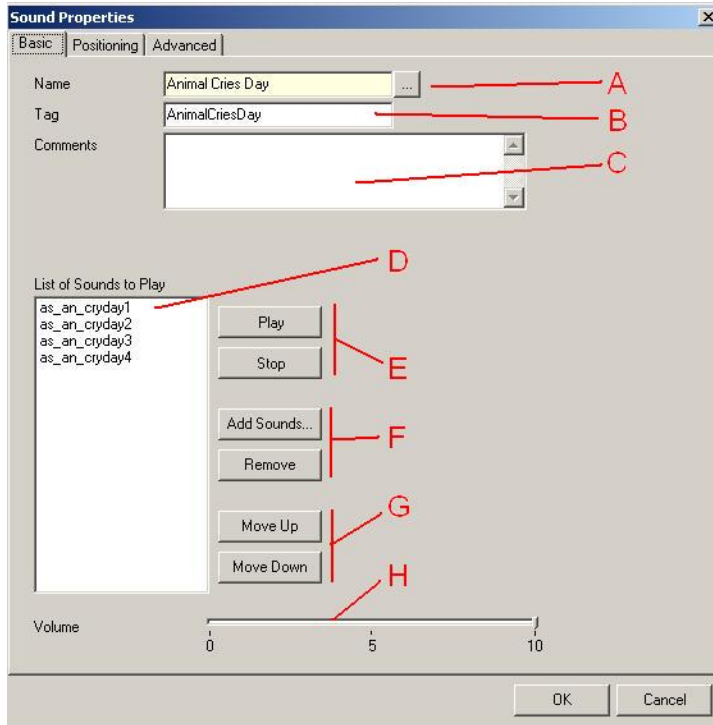
11.2 The Properties of a Sound Object

You will probably want to adjust the properties of every sound object you place in your module, even the pre-generated ones included in the standard palette. The reason for this is quite simple: all sound objects in the palette have the exact same properties (with a few exceptions) and play in exactly the same fashion. The only thing that changes is the list of sounds that are included in the sound object. Fortunately, editing the properties of these objects is quite simple.

11.2.1 The Basic Properties of a Sound Object

Editing the basic properties of a sound object can be a long process. Not because the information provided in this property screen is complicated, but because it can take some time for the toolset to add new sounds to your list of sounds to play.

Image 11.2.1a – Sound Properties Screen – Basic Tab



- A** Name: The name of a sound object is for reference purposes only since players will never see that name.
- B** Tag: The tag of a sound object is quite important when the sound is not active or when it is played only through a script since it is by its tag that you will identify it in a script. Please refer to Sub-Section 11.2.3 for more details on inactive sound objects.
- C** Comments: As for any other object, the “Comments” section is for you to use at will. You can leave here any comments or notes that you might find useful if you have to further edit the sound object at a later date.
- D** List of Sounds to Play: This is all the sounds that are included in your sound object. The properties setting whether these sounds are played in this specific order or randomly, and the timing which controls when these sounds will be played are all editable under the “Advanced” tab of the Sound Properties screen.
- E** Play/Stop Buttons: Clicking on these buttons permits you to hear the sounds that are in the list of sounds, and therefore to determine if you want to keep or remove them.
- F** Add Sounds.../Remove Buttons: These buttons permit you to add or remove sounds from the list of sounds to play. When you hit the “Add Sounds...” button, you will open the Sound Resources Screen (as shown in Image 11.2.1b below).

The “Sound Resources Screen” may take several seconds to open due to the huge amount of available sounds. Fortunately, you can test the sounds available in the resources screen by clicking on the “Play” button on top of the screen, which will help you to avoid adding sounds that you do not want to keep.

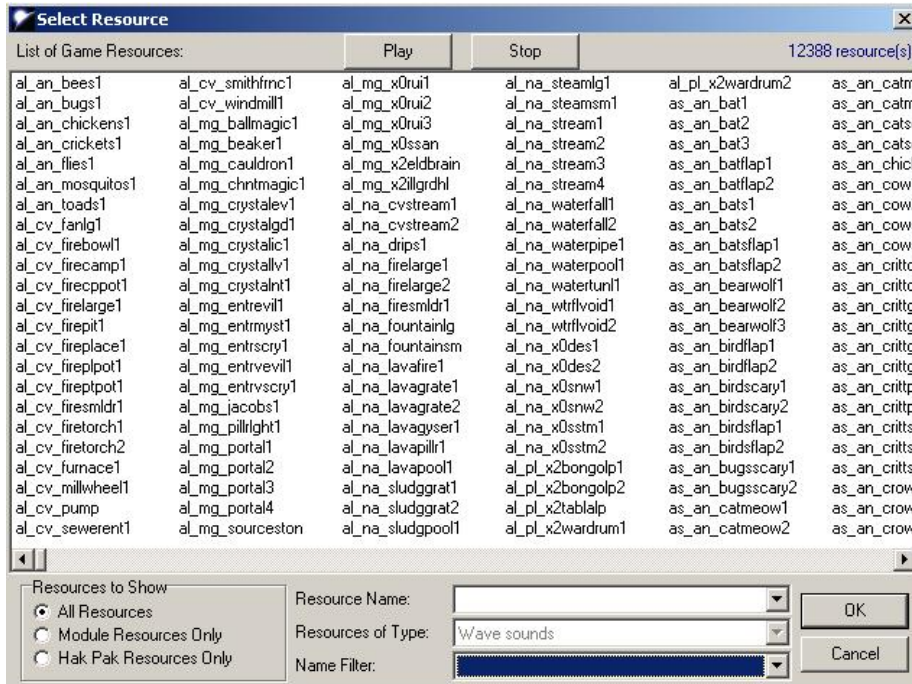
You will notice that the names of the available sound resources are categorized by their prefixes. The first letter identifies what type of sound this is (“a” for ambient, “c” for creature, “fs” for footsteps, “vs” for voice set). Knowing these prefixes is not necessary since you probably want to use the filters that are available to limit the list of resources, while on the other hand some of these prefixes are used in the list of filters as well.

Filters can be added by using the dropdown menu named “Name Filter” at the bottom of the resources screen. If you are looking for ambient sounds, you should know that “as” at the beginning of the ambient sound filters means the ambient sounds that are meant to be played as a single shot (like the sound of a bell) while “al” means the ambient sound that are meant to be played as a loop (like the sound made by fire).

Once you have applied the desired filter, finding the right sound may still seem daunting depending on how many sounds are covered by this filter. To help you further, you may type in the “Resource Name” field a word that describes the sound you are looking for. We suggest you to type only the first three letters just to be sure to catch any abbreviation of this word. Bioware has fortunately well-named their sound files so you should be able to catch all the sounds that fit your criteria using that technique however you may want to try several different descriptive words to increase the chance of locating just the right sound.

- G** Move Up/Move Down Buttons: These buttons permit you to modify the order in which the sounds appear in the list. The order is only important if you set your sound object to play the sounds in the order they appear in the list (which can be set under the “Advanced” tab).
- H** Volume: The overall volume of the game (sounds and music) is set by the players. Therefore, setting the volume of a sound object is totally relative. The volume setting only establishes how loud the sound object will play compared to the other sounds.

Image 11.2.1b – Sound Resources Screen

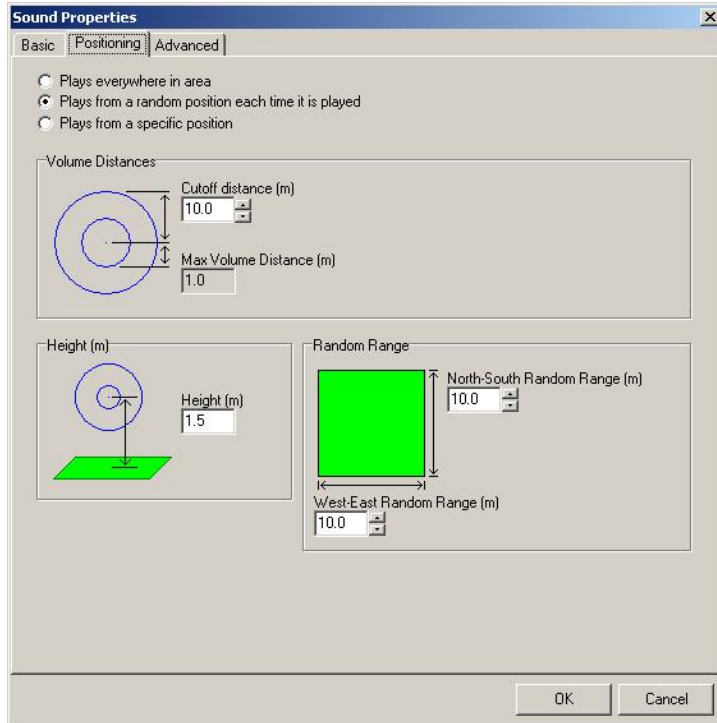


11.2.2 The Positioning of a Sound Object

The location of a sound object can be adjusted directly in the toolset window by clicking and dragging the sound where you want it to be located. However, if you want to change how far away you want players to be able to hear your sound, then this is where to look.

You will notice that all distances here are calculated in meters. Note that the size of a tile is 10 x 10 meters.

Image 11.2.2 – Sound Properties Screen – Positioning Tab



Plays everywhere / from a random position / from a specific position: This setting is self-explanatory. You might want to have sound objects that are ambient and not specific to a tile or a feature of your area be played “everywhere in area”. This will avoid the requirement to duplicate this sound many times in the same area.

Volume Distances: If your sound is not set to be played everywhere in the area, you can adjust here at what distance (in meters) of its actual location the sound can be heard by players. The “Max Volume Distance” is always 10% of the “Cutoff distance” of the sound object. Once players are beyond the “Max Volume Distance”, the sound volume will be reduced proportionally to the distance players are from the center of the sound. You should be aware that, unless players adjust their sound settings to hear them very loudly, a sound will become almost inaudible well before the players have exited the “Cutoff distance” zone. You might therefore want to make the “Max Volume Distance” very wide if you want players to be able to hear the sound far from its actual location.

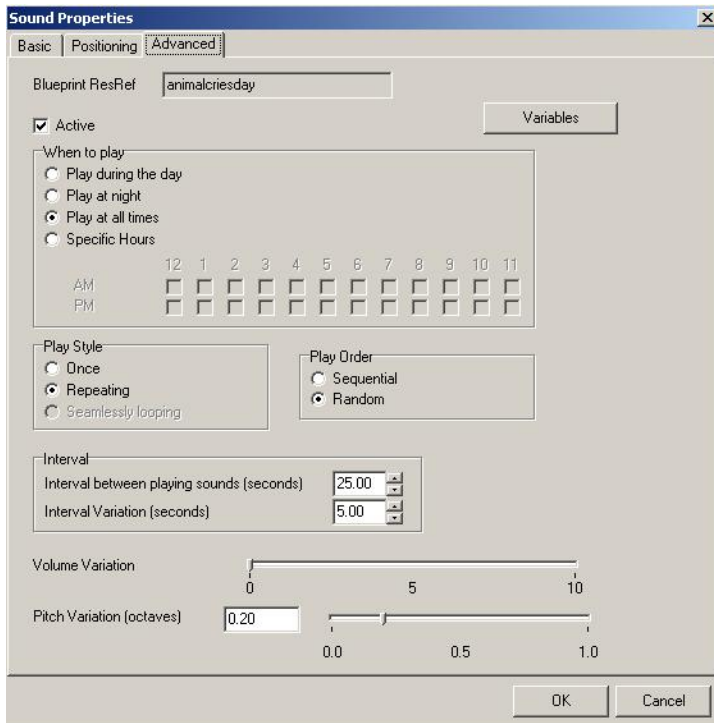
Height: If your sound is not set to be played everywhere in the area, you can adjust the height of the source of the sound. The default height of 1.5 meters is basically the average height of player characters. You should not change this parameter if you want to be sure that players will be able to hear your sound.

Random Range: If your sound is set to be played from a random position each time it is played, you can determine here how wide (in meters) the random zone in which the sound can be played is. The location where you placed your sound object in the area is in the middle of this large square.

11.2.3 The Advanced Properties of a Sound Object

The advanced properties of a sound object will permit you to determine when and how the sounds included in your object will be played.

Image 11.2.3 – Sound Properties Screen – Advanced Tab



Blueprint ResRef: The blueprint ResRef is the unique reference identifying the sound object blueprint from which this sound object was created. This reference is mainly useful when you want to update all sound objects already placed in your module in accordance with a modification you have made to the blueprint that was used to create them (this can be done by right-clicking on the blueprint and selecting the “Update Instances” in the dropdown menu).

Active Box: An active sound object will play its sounds according to its selected properties. If you untick this box, then the sound will be inactive and will therefore not play until it is activated. To activate a sound object during gameplay, you must run a script that includes the following commands:

```
void main()
{
object oSound = GetObjectByTag("ADD_TAG_OF_SOUND_OBJECT");
SoundObjectPlay(oSound);
}
```

Where you run this script depends on what event triggers the requirement to play the sound. To disable a sound object, replace the SoundObjectPlay() command in this script with SoundObjectStop().

Variables: The “Variables” assigned to a sound object may be used in a script, but otherwise they do not have any effect.

“When to Play” settings: These settings are self-explanatory.

“Play Style” settings: The sound object can either be played once, in a repeating fashion (in the interval set in the “Interval” settings below) or seamlessly looping. A sound object can play seamlessly looping only if there is only one sound in the sound list under the “Basic” tab.

If a sound object is set to be played only once, you will need to tell it when to play through a script. The script to be used in for that purpose is the same as the one mentioned above to have an inactive sound start playing. The sound will then be played once only and will not play again until this script is called again for it.

“Play Order” settings: If you set the “Play Order” property to sequential, the sounds listed under the “Basic” tab of the sound properties will be played in the order they are listed. Otherwise, they will be played in a random order.

“Interval” settings: If the sound object is set to be played “Repeating” in the “Play Style” setting, you can determine exactly when you expect the sound to be played again. Do not forget that if all your sound objects have the same interval, they will be heard almost all together, followed by a period of silence. It is therefore a good idea to add some variety in the intervals you give to your sound objects placed in an area.

Volume/Pitch Variation: These settings allow you to add variation to the volume and pitch of the sounds that are played, except when they are set to be played seamlessly looping.

11.3 Designing Tips

Do not go overboard in adding sounds to your areas: Sounds are very effective in making your area coming alive. On the other hand, it is quite easy to make an area unrealistic while you are adding sounds. Does the bell of the church near your home rings every 25 seconds? Do you hear owls every 12 seconds when you walk in a forest? Probably not. Sounds are nice, but too many sounds or repetition of the same sounds over and over again may become a flaw in your module.

11.4 How to...

11.4.1 ...make a creature play a sound

This is not done through a sound object, but rather through a script. The scripting command you are looking for to accomplish this is "PlaySound()", which should be placed in a script and the script placed in an appropriate event of the creature you want to play the sound. The sound name that must be used with this command is the same as those listed in the Sounds Resources Screen as seen above.

CHAPTER 12 Triggers

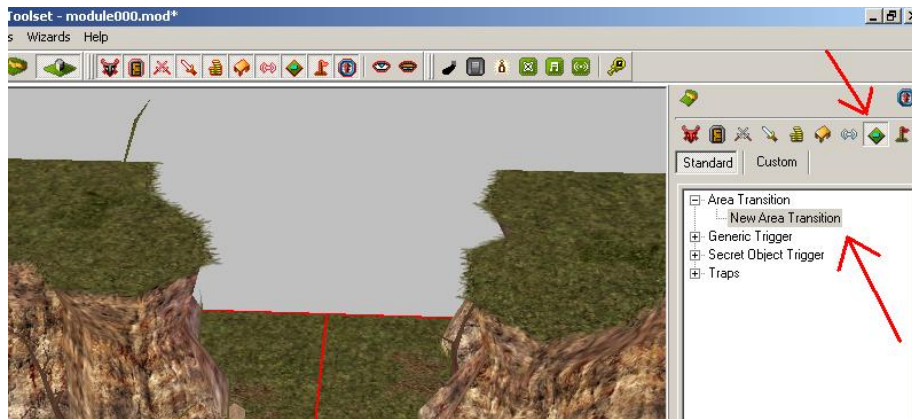
Trigger objects serve many purposes. Basically a trigger is a zone that you paint in your area for the purpose of initiating an event when players (or any other creatures) enter or exit its boundaries.

Encounter objects are triggers as well, but they are isolated in a different category in the toolset, probably because it was more practical or ergonomic that way. You will however find that many of the comments made in this Chapter have also been made in Chapter 10 covering encounter objects.

12.1 Adding a Trigger

Adding a trigger to your module is simply a matter of painting it in the location you wish to be covered by its effects.

Image 12.1 – Adding a Trigger



To that end, you need to:

- 1) Select the appropriate trigger blueprint in the trigger palette as shown in the image above;
- 2) Determine the boundaries of the zone you wish to be covered by the trigger by using your cursor as a paintbrush;
- 3) Double-click to close the zone when you have properly painted the boundaries of your trigger.

If the terrain type in your area is not flat, you will notice that your trigger will either disappear below the terrain level or be fixed in the air. To avoid this, click on your mouse more often to set down more points for your boundaries so that the trigger, when you close it, will take into consideration the shape of the terrain.

12.2 Categories of Trigger

When you expand your trigger blueprint palette, you will notice that trigger objects are classified in four different categories:

- i) Area Transition Triggers;
- ii) Generic Triggers;
- iii) Secret Object Triggers; and
- iv) Trap Triggers.

A trigger that is an Area Transition will show up in the game as being these large purplish zones on the ground that, when you click on them, bring you to another area. There is only one purpose for this trigger type, to jump players to another location.

A Generic Trigger is basically an object that you paint on the ground that has no other effects but those created by scripts you attach to it.

Secret Object Triggers determine, when a player enters them, if a successful search skill check has been made by the player and, if so, reveal something that was first hidden. In fact, Secret Object Triggers are generic triggers with pre-made scripts added, they do not have any other particular properties.

Trap triggers are traps that are set on the ground and not on a door or placeable object.

Each one of these trigger categories will be covered in detailed below.

12.3 Creating a New Trigger Blueprint

There is no real reason to create new trigger blueprints since trigger objects are pretty generic and there is not much customization to be done that would justify creating a new blueprint. The existing blueprints in the Standard palette will probably satisfy all your needs. Unless you have a special purpose trigger that you intend to use in several locations

If you still need to create a new blueprint, as an example if you often use the same design for a trap and you are tired of editing the standard trap blueprint you are using, we recommend right-clicking on your custom trigger and choosing the “Add to Palette” option in the dropdown menu.

Another way to create a new blueprint would be to use the Trigger Wizard (in the Wizards menu of the toolset). We do not recommend using this technique since this wizard does not do anything for you, therefore we are not going to cover this wizard in at this time. If enough people request it we may expand this chapter in the future

12.4 The Properties of a Trigger Object

All four categories of trigger objects have many properties in common and they are organized in the same way. The properties that are specific to some categories of trigger object will be covered below in a distinct section. Note however that all these “specific” properties are listed under an

additional tab and therefore the properties listed under the “Basic”, “Scripts”, “Advanced” and “Comments” tabs are all exactly the same for every category of trigger object.

12.4.1 The Basic Properties of a Trigger Object

Image 12.4.1 – Trigger Properties – Basic Tab



The Basic Properties of every category of Trigger Object are those that will permit you to identify and categorize the trigger.

Name: The name that you will give to your trigger object is for reference purpose only. You will notice that trigger objects are listed in your Module Structure Window by their tag and not by their name.

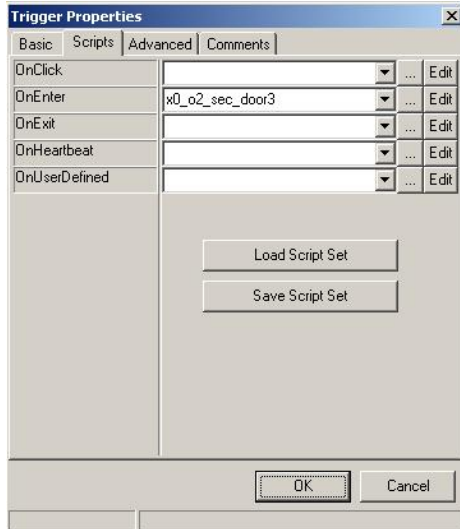
Tag: The tag permits you to identify the trigger through a script.

Trigger Type: This permits you to identify which type of trigger your object is. Keep in mind that, as mentioned previously, Secret Object Trigger blueprints are in fact generic triggers with scripts that were added by Bioware in the blueprint.

12.4.2 The Script Events of a Trigger Object

All trigger objects have the same 5 script events, but only the Secret Object Trigger comes with pre-made scripts.

Image 12.4.2 – Trigger Properties – Scripts Tab



OnClick: This event will fire when a player clicks on the trigger object. However, only Area Transition triggers are “clickable”, so this event will not be of any use for other trigger objects. The OnClick event on an area transition can be used if you do not wish to set the area transition (i.e. the jumping of players) through the properties designed for that purpose or if you want to add further events in addition to the result that players will be sent to another location when they click on the trigger.

OnEnter: This event will fire whenever a player or a creature enters the zone that is covered by the trigger you have painted.

You will notice that all Secret Object Triggers come with a script placed in that event. This is the script which checks if the conditions are met to have the

secret object appear. The only way for you to see what those conditions are and what will then appear is by clicking on the “Edit” button besides this script and read the comments placed at the beginning of the script by Bioware.

OnExit: This script will fire whenever a player or a creature exits the zone that is covered by the trigger you have painted.

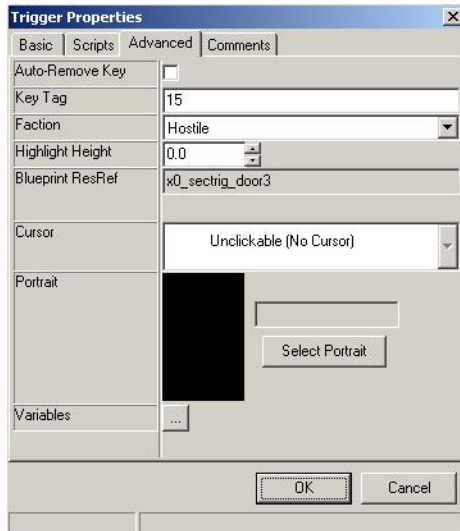
OnHeartbeat: The script associated with this event fires every six seconds. This is good for a script that needs to check for something repeatedly or perform an action regularly. Be aware that too many OnHeartbeat scripts can be processor costly. Use other types of event scripts when possible.

OnUserDefined: The script attached to this event fires whenever an event is signalled to the object. This requires intermediate scripting knowledge. Typically this script is a switch/case statement that does something different depending on the signal (numerical value) sent to it by an event that calls the script.

12.4.3 The Advanced Properties of a Trigger Object

All categories of trigger objects have the exact same advanced properties, although they might not all be relevant to each one of these categories.

Image 12.4.3 – Trigger Properties – Advanced Tab



Auto-Remove Key: Only Area Transition triggers may require a key to be activated. This can be done by adding the item's tag (the item that a player needs to have in its inventory to activate the area transition, in other words the key) to the Key Tag property (see below). If you want the item to be removed from the player's inventory after the area transition has been used, tick this box.

Key Tag: If you want players to be able to use an area transition trigger only if they have a certain item in their inventory, place the tag of the item in this spot.

When your trigger is a Secret Object trigger, the number appearing in the Key Tag property is not related at all to a key. It is the difficulty challenge (i.e. DC) of the search skill check that is performed

to determine if they have discovered the secret object when players enter the trigger.

Faction: This property is relevant only for traps. For a player to trigger a trap, the trap needs to be set to a faction that is hostile to the player.

Highlight Height: This property is relevant only to area transition triggers. As you know, area transition triggers are painted on the ground and they will appear on the ground to players. It however happens sometimes that the ground is not flat and therefore you do not see the trigger very well. Increasing the "Highlight Height" value allows players to see the area transition even on the parts that have been painted on terrain that is higher than the ground level. This property however has its limits since, for players to be able to see the trigger, they still have to scroll the mouse on the ground level itself or press the "Tab" key to make the trigger light up.

Blueprint ResRef: The blueprint ResRef is the unique reference identifying the trigger blueprint from which this trigger object was created. This reference is mainly useful when you want to update all triggers already placed in your module in accordance with a modification you have made to the blueprint that was used to create them (this can be done by right-clicking on the blueprint and selecting the "Update Instances" in the dropdown menu).

Cursor: This property is relevant only for area transition triggers. It determines how the cursor will look like when players scroll their cursor over the area transition. Unfortunately, there is only one cursor appearance available so this property is basically useless.

Portrait: You can add a portrait to your trigger. But since a trigger object does not have any conversation file and cannot speak any string, even through a script, we have yet to find a use of adding a portrait to a trigger.

Variables: The “Variables” assigned to a trigger may be used in a script, but otherwise they do not have any effect.

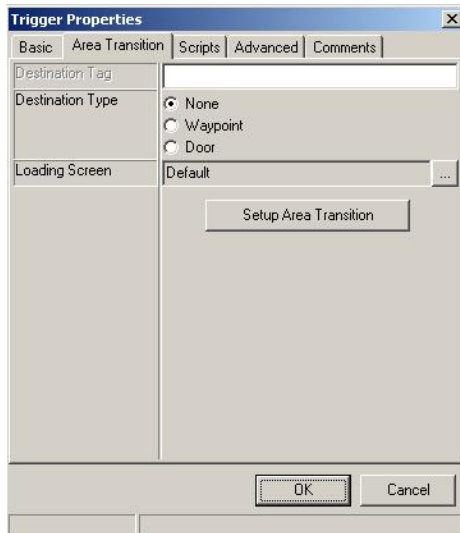
12.4.4 The Comments on a Trigger Object

The “Comments” tab on a trigger will provide you with a large text window in which you can write anything you like (notes about the purpose of the trigger or anything else that can be useful to the builder). You will also notice that every standard trap blueprint has the details about the damages that will be made to players that trigger the trap in the comments, which quite useful to help properly balance your module.

12.5 Properties Specific to Area Transitions Triggers

Area transition triggers are used as a means to transition from a location to another one. Players use them by clicking on the light-blue polygon that appears when they scroll their cursor over the area transition.

Image 12.5 – Trigger Properties – Area Transition Tab



To activate an area transition, you need to assign a unique tag to your trigger under the “Basic” tab. Then click on the “Setup Area Transition” that appears under the “Area Transition” tab of your trigger’s properties.

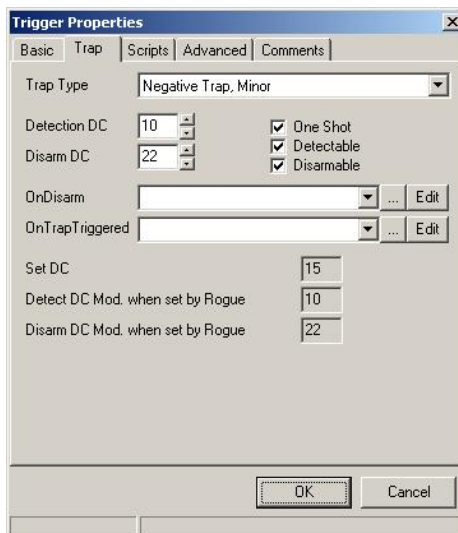
We have already commented in detail on the process that needs to be followed to add a transition between doors under Sub-Section 9.1.4. It is basically the same for area transitions. We therefore refer you to this Sub-Section for more comments on that subject.

When a transition is added to an area transition, the toolset automatically creates a waypoint right in the middle of your trigger. This waypoint used by the engine to determine where the players will actually appear when they use the

transition that leads to your area transition (if you activated it as a “both ways” transition). Pay a special attention to this waypoint, since it also determines which direction players will be facing after they use the transition. When the area transition is generated, the waypoint is automatically facing north. You may want to rotate it so that players using the area transition face the proper direction when they appear on the location of the waypoint.

12.6 Properties Specific to Trap Triggers

Image 12.6 – Trigger Properties – Trap Tab



When the trigger that you have painted is a trap, you will notice that an additional tab identified as "Trap" as been added to its property screen. Under this tab you will find all the properties that are relevant to the trap that you have set. These properties are identical to those of a trap set on a placeable object, which are fully discussed in Sub-Section 8.3.3 of this Manual. Refer you to this Sub-Section for more details concerning these properties.

As mentioned previously, the damage that is being applied to any creature that has triggered a trap trigger is detailed under the "Comments" tab if you have used the standard trap blueprints. Note that these comments will not be updated if you change the trap type of your trigger.

12.7 Built-in Generic Triggers

When you expand the "Generic Trigger" list of the trigger palette, will find many blueprints are listed there. Some of these blueprints may be of interest, but those that are require a bit of customizing.

12.7.1 Henchman Trigger

This is a trigger to signal henchman user defined events. These events were coded in the SoU official campaign modules and generally resulted in triggering conversations between the player and his henchman.

As mentioned in the comments of this trigger blueprint, this trigger only works with the SoU modules and therefore is of absolutely no interest to us. This blueprint should definitely be removed from the palette by Bioware, but there is nothing we can do about it.

12.7.2 Tracks Trigger

Why was this trigger called "Tracks Trigger"? We have no idea. The purpose of this trigger is to make any player character enters the trigger speak a one-liner conversation. This one-liner will be spoken only once, but by every player character enters the trigger.

In order to make this trigger work, you must:

- 3) Paint the trigger where you want the player characters to speak the one-liner;

- 4) Give to the trigger a unique tag;
- 5) Create a conversation file with only one node, which will be the line you want player characters to speak;
- 6) Save your conversation file by giving it the exact same name (respecting the case) as the unique tag you have assigned to the trigger object.

12.7.3 X0_SAFEREST Trigger

This is the trigger that was used in the SoU expansion to designate the small rooms in which, while their door was closed, players could rest while otherwise no rest was allowed in the area. This trigger needs the SoU modules' OnRest scripts and some customization on area tags. We do not recommend using this trigger since the SoU scripts involved are too "SoU related" and you are probably better off creating your own rest restriction scripts if you want to create a resting system that is similar to the one used in the SoU expansion.

12.7.4 XP2 Interjection/One-Liner, Non-Random/Random Triggers

These triggers were used in the HotU expansion to have the henchmen engage in conversation or speak on-liners. These triggers were really designed for the HotU campaign and they were not adjusted so that builders could easily use them in their own modules.

In any event, Michael "Huntsman" Paddon has written a great tutorial that provides you with all the details on how to use these triggers, which can be found here:

<http://www.btinternet.com/~huntsmans/Huntsman2.htm>

12.8 Designing Tips

Make sure that all the waypoints created for your area transition triggers are facing in the right direction: As mentioned previously, when you create an area transition trigger the toolset creates a waypoint in the middle of the area transition to identify the spot where players will land when they will use the transition. By default, these waypoints are facing north, and so will the players when they use the area transition. Rotate these waypoints so that they point in the direction in which it is logical for the incoming players to face when they enter the area and the problem will be solved.

To rotate a waypoint right-click on the waypoint and select 'adjust location' from the drop down menu. Adjust the Bearing of the waypoint by increasing or decreasing the gradient (numbers) in relation to its current position.

Do not over-use the trap triggers: Unless a module is designed for rogue characters, do not forget that all other character types will not be very good at finding traps and, on top of that, they will most probably have no way to remove them. Most non-rogue characters will usually therefore simply walk right through most traps, which can be deadly and very frustrating if you over-use them. If you add a trap trigger every 5 meters in a pathway that cannot be avoided, you will anger all players that do not play a rogue (and even rogues will eventually get frustrated with this).

Do not over use secret objects and give hints for those secret objects that are plot related: Again, unless a module is designed for rogue characters, you can not expect all characters to have boosted search skills and many will not find your secret objects. If a secret door is vital to the plot (i.e. the module cannot be finished unless this secret door is found and used by players), make it easy to find for all character types. Add hints, like tracks on the floor and conversation that refers to people suddenly disappearing near a wall, so that barbarians know where to look for the secret door.

12.9 How to...

12.9.1 ...make the Secret Object Triggers work

- 1) Draw the trigger that is named appropriately for the kind of secret door you wish to create. Do not forget that a search skill check will be made when a player enters the trigger, so when you draw the boundaries of the trigger you should take into consideration that they should not be too wide (a player shouldn't be able to spot a secret door while they do not even see the wall on which the door will appear) or too narrow.
- 2) Give a unique tag to your trigger.
- 3) Place a waypoint object on the location where you want the secret door to be revealed. Change the tag of this waypoint to LOC_<tag of the secret door trigger>.
- 4) If the secret object involves some kind of transportation (i.e. a door, a portal, a trap door, etc.), place another waypoint object on the location where the secret door leads. Change the tag of this waypoint for DST_<tag of the secret door trigger>.
- 5) Adjust the "Key Tag" property, in the advanced properties of your trigger, to the appropriate difficulty check roll you wish to set for the players to be able to spot the secret object on a search skill check.

12.9.2 ...make a custom placeable object appear while using a Secret Object Trigger

Scripts associated with Secret Object Triggers are fortunately quite easy to customize. In order to change the standard "Secret Object" that is under a trigger to a custom one:

- 1) Create a new placeable object blueprint for the custom object you wish to appear instead of the standard one;
- 2) Paint any Secret Object Trigger;
- 3) Give a unique tag to your trigger;
- 4) Place a waypoint object on the location where you want the secret door to be revealed. Change the tag of this waypoint for LOC_<tag of the secret door trigger>.
- 5) Under the "Scripts" tab of your trigger's properties, click on the "edit" button next to the OnEnter script event of the trigger to edit the standard script associated with the trigger. In this script, there will be a line near the end that will read:

```
RevealSecretItem("x0_**something**");
```

Change the string that is between the "" for the blueprint ResRef of your custom placeable object. Save this script by giving it a new and unique name using the Save As command.

- 6) Adjust the "Key Tag" property, in the advanced properties of your trigger, to the appropriate difficulty check roll you wish to set for the players to be able to spot the secret object on a search skill check.

12.9.3 ...have an area transition trigger appear in game even though no actual area transition event has been assigned to it

Create your area transition trigger as usual and, under the "Area Transition" tab of its properties, make the "Destination Type" a waypoint but do not add any value in the "Destination Tag" and do not click on the "Setup Area Transition" button. The area transition will then appear during gameplay but will lead to nowhere when a player clicks on it. You may then add any script you want in the OnClick event (under the "Scripts" tab of the trigger's properties) to make something happen when players click on the trigger.

This technique is commonly used when you want to add an area transition on an open border of an area but you do not want players to be able to use it because, as an example, the road would simply lead them out of the scope of the module. In such an example, the script to be added to the OnClick event of the trigger would look something like this:

```
void main()
{
    object oPC = GetClickingObject();
    SendMessageToPC(oPC, "This road leads to a region that is not of any interest right now.");
}
```

CHAPTER 13 Waypoints

Waypoint objects have one use: to mark a location. They have no other purpose, but still they are really useful and often are the most frequently used objects in a module.

From the moment you want a creature to move from one point to another, to have players sent to a location, to create an object or any other action or command that requires you to identify a location in your module, you will need to use a waypoint.

Placing waypoint objects is as simple as placing any other object. Waypoints have their own palette category. When you expand the “Waypoint” menu, a list of waypoint blueprints will appear. The common waypoint is listed under the surprising name of “Waypoint”.

13.1 Creating a New Waypoint Blueprint

As mentioned above, a waypoint’s only purpose is to identify a location. For that purpose, a waypoint object only requires very limited properties and the standard “Waypoint” blueprint should normally satisfy your needs.

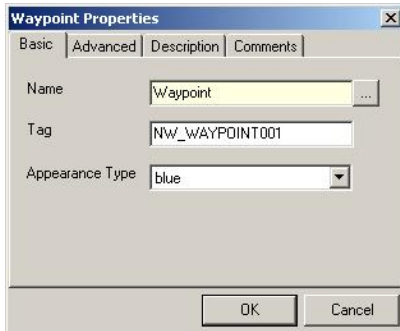
If you however often make the same modifications to the standard waypoint objects you place and you therefore feel the need to create a new blueprint to the Waypoint Palette, right-click on your custom waypoint that you wish to use as a blueprint and select the “Add to Palette” option in the dropdown menu. The new blueprint will then appear in the Custom waypoints palette.

13.2 The Properties of a Waypoint

The properties of a waypoint are very limited. Of course, this is due to the fact that a waypoint is only used to identify a location and do nothing on its own (except identifying map notes). But still, it is very important to edit the properties of your waypoints since you will want to assign them unique tags.

13.2.1 The Basic Properties of a Waypoint

Image 13.2.1 – Waypoint Properties– Basic Tab



Name: The name of a waypoint is for reference purposes and does not have any use. You may even notice that waypoints are listed by their tag in the Module Structure Tree, not by their name.

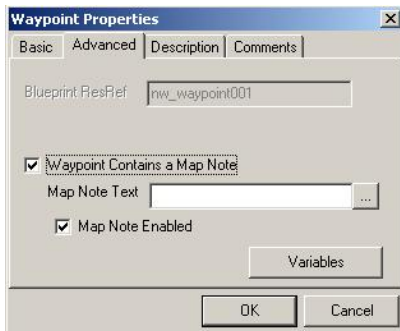
Tag: The tag of a waypoint is of great importance. As already mentioned, a waypoint is used to identify a location. To identify the location, the engine must identify the waypoint and this is done by looking for the waypoint having a specific tag. Therefore, to avoid duplicating locations, you will generally want to assign

unique tags to your waypoints.

Appearance Type: Your toolset screen can rapidly become full of waypoints in populated areas. In order to visually sort certain categories of waypoints from others, you can change their appearance type here. A waypoint will always be a yellow arrow, but each flag colour that can be selected here comes with a different shape for the yellow arrow.

13.2.2 The Advanced Properties of a Waypoint

Image 13.2.2 – Waypoint Properties– Advanced Tab



Blueprint ResRef: The blueprint ResRef is the unique reference identifying the waypoint blueprint from which this waypoint was created. This unique reference must be used when you want to create a waypoint through a script.

Waypoint Contains a Map Note: If you tick this box, then a map note will be assigned to this waypoint. Map notes are those little stars that can be seen by players which show a point of interest on the map when their map mode is activated.

Map Note Text: This is the text that players will see when they will scroll their cursor over the map note.

Map Note Enabled: This confirms that the map note contained in the waypoint can be seen by players. There might be notes that you do not want players to see up front, but only after they have discovered something specific, after a quest was assigned to them or simply after they were given directions on how to go to the location of this waypoint. If that is the case, you can disable the map note by unchecking this box. The map note text will still be included in the waypoint, but the map note

will be inactive. In order to activate later on, place the following script in the script event where you want this information to be revealed to the player:

```
void main()
{
    object oWP = GetWaypointByTag("ADD_TAG_OF_WP");
    SetMapPinEnabled(oWP, TRUE);
}
```

Variables: The “Variables” assigned to a waypoint may be used in a script, but otherwise they do not have any effect on the waypoint itself. There is one variable that can be assigned to a waypoint object for which the script has already been written in the game, which is:

X2_L_WAYPOINT_SETFACING

If you add this variable as an integer to a waypoint object that is part of a WalkWayPoints routine of a creature and if you set its value to 1, the creature will assume the facing of the waypoint when it reaches it before it walks to the other waypoints of the routine. This can be useful in controlling the behaviour of patrolling guards or similar situations.

13.3 Built-in Waypoint Blueprints

The waypoint palette has many other waypoints besides the standard “Waypoint” object. To know more about why and how to use these waypoints, place one in your area and open its properties. You will find detailed instructions on how to use them under the “Comments” tab of these built-in waypoints.

13.4 Designing Tips

Add map notes to designate any points of interest and the area transition locations in each of your areas: True, it does not make sense for a player to know where the house of the gardener is located before the player actually goes there. However, you must never forget that playing your module is supposed to be fun. Being lost all the time because you cannot find the places where you need to go is not fun and you must not take for granted that players will be keeping notes. The only time where you should not identify a location or an area transition with a map note is when finding this location is actually part of the quest.

13.5 How to...

13.5.1 ...have a creature walk amongst waypoints

- 1) Give a unique tag to your creature.
- 2) Make sure that you have not removed the WalkWayPoints() command of the default OnSpawn script of the creature (if you have not changed the OnSpawn script event of the creature in its properties, then nothing needs to be done here).

- 3) Place the creature in the area where you want it to perform its walking routine.
- 4) Select the creature by clicking on it.
- 5) While the creature is still selected, right-click on the locations in the area where you want its walking routine to lead it and choose the "Create Waypoint" option in the popup menu that appears. You may add as many waypoints as you want to create its walking routine.

A white line will appear on the ground with your creature at its origin. This line shows you the path that the creature will be following in its walking routine. Avoid place the waypoints in such a way that the lines drawn cross over impassable objects or tiles as this may cause the creature to lag or experience pathing problems. It is highly recommended that you add waypoints that will aid the creature in maneuvering around obstacles. This will reduce any possible pathing related issues.

NOTE: Keep in mind that creatures will always walk their waypoints from waypoint 1 to 2 to 3 to 4 to ... and back again from ... 4 to 3 to 2 to 1. Creatures, by default, will not walk from the last waypoint to the first (e.g. from 4 to 1) but will reverse their steps (1>2>3>4>3>2>1).

13.5.2 ...make walking routines cross an area transitions

For creatures that were assigned a walking routine to be able to use area transitions (doors or area transition triggers), you need to assign a variable called X2_SWITCH_CROSSAREA_WALKWAYPOINTS as an integer on the module itself and give it a value of 1. See Sub-Section 2.2.3 for more details about module variables.

CHAPTER 14 Merchants

A merchant in the toolset is not the creature that is selling items, but rather the actual “store” that cannot be seen by players. It has all the information related to the items that can be bought or sold, the mark up prices of these items, and any other relevant information related to the action of trading that is involved when a player buys or sells an item to an NPC.

Of course, Merchant Objects are only an accessory to a creature. Without a creature that offers to buy or sell items to player characters, the Merchant Object or store is of no use at all. It will therefore be important to link your Merchant Object to the NPC that will be operating the store.

14.1 Creating a New Merchant Blueprint

The Merchant Wizard that is available in the toolset is a 2-step wizard. You first determine where the blueprint will appear in the merchant palette and then, as a second step, you give a name to the merchant object. This of course is of no use at all and we rather recommend you use the Store Setup Wizard (that is fully commented in Section 14.2 below) to create a new merchant object instead of taking the long way by first creating a new merchant blueprint.

14.2 How to Turn a Creature into a Storekeeper

The Store Setup Wizard, probably the best wizard available in the toolset, permits you to do this easily. In order to launch this wizard, place the creature you wish to act as a storekeeper in an area, right-click on it and select the “Setup Store” option in the dropdown menu.

The Store Setup Wizard will then bring you into the following 3-step process.

Image 14.2a – Store Setup Wizard – Step 1

Store Setup Wizard

Conversation

Enter the conversation text for the shopkeeper.

What does the shopkeeper say when the conversation begins?
Welcome to my shop. Would you like to see my wares?

What does the player say when the player wishes to see the store?
Yes. I'd like to see what you have for sale.

What does the player say when the player is not interested?
No, thanks.

Use appraise checks

(optional) Enter the filename for the shopkeeper's conversation
store001

(optional) Enter the filename for the script that will open the store
openstore001

< Back **Next >** Finish Cancel

STEP 1: The wizard first creates a very basic conversation that will be assigned to your storekeeper. The information/instructions to that effect are quite straightforward and do not need further comments. Note that you will be able to edit this conversation later on, as for any other conversation file, and add more substance to it.

You will also need to tick the “Use appraise checks” if you want the appraisal skill of players to be taken into consideration while they do their shopping.

Finally, although it is mentioned that this is optional, you need to give a name to the conversation file and the script file that is being created by the wizard to set up the store. If you are following any naming convention, you should follow it here as well since these files will be appearing in your module’s script and conversation resource lists.

Image 14.2b – Store Setup Wizard – Step 2

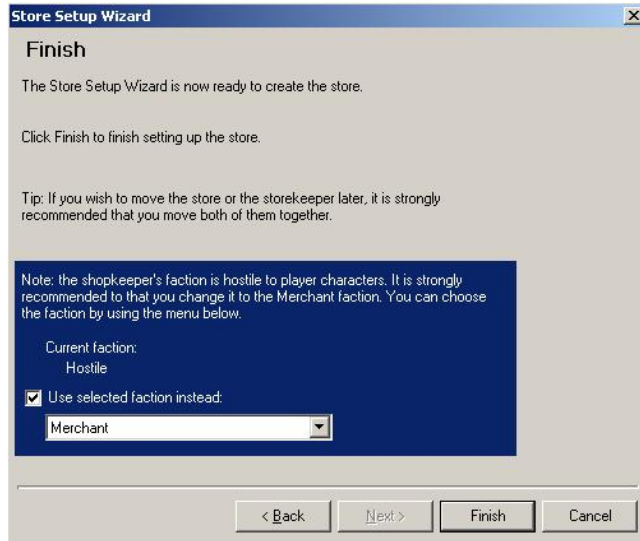


STEP 2: You then need to determine what kind of store your storekeeper will be operating. As previously mentioned, it is not really useful to create custom merchant blueprints, so you will probably need to select one of the standard merchant blueprints in the palette that appears in this screen. These blueprints are all used in the original campaign of NWN and its two expansions.

Many of these blueprints have the exact same name. For these blueprints that have the same name, you can take for granted that whichever one comes first in the list has stuff available for sale for lower level characters, the next one has better stuff and so on and so on. You can also take for granted that the name of these blueprints are representative of what is being sold by these stores.

The inventory available for sale is customizable at a later stage.

Image 14.2c – Store Setup Wizard – Step 3



STEP 3: The final step, nothing needs to be done here unless you were as negligent as we were and have created a storekeeper with a creature that is a member of a faction that is hostile to player characters.

14.3 The Properties of a Merchant Object

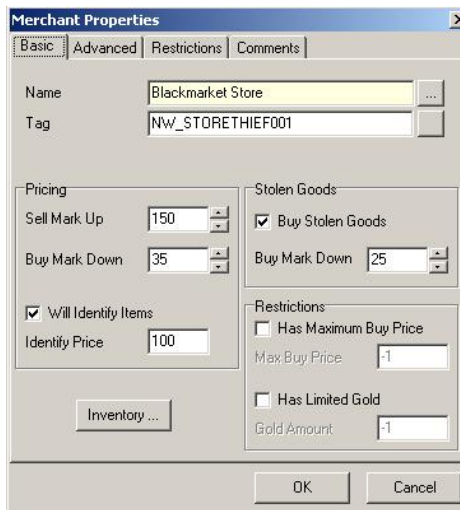
Once you have set up a store and linked it to your storekeeper, you may adjust the properties of the merchant object to determine the conditions under which a trade can be made and also what exactly can be traded by the storekeeper.

To access the properties of the merchant object, right-click on it and choose the "Properties" option in the dropdown menu. The merchant object in the toolset looks exactly the same as a waypoint (i.e. a yellow arrow) and was placed on the location of your storekeeper in the setup process.

14.3.1 The Basic Properties of a Merchant Object

The basic properties of a store are those that determine the economical terms and conditions that are followed by the storekeeper to operate the store.

Image 14.3.1a – Merchant Properties Screen – Basic Tab



Name: You can give any name to your merchant object, this name will not appear in the game and is for reference purposes only.

Tag: Do not change the tag of a merchant object unless it is absolutely necessary. If you need to change this tag, you will have to edit the script that was created in the store setup process to update the tag reference made in this script as well.

Pricing: The price of any item is determined in the item's general properties. (See Sub-Section 7.3.1.) Of course, a storekeeper needs to make a profit so he will not buy and sell items at their actual value. These pricing properties allow you to determine the mark up / mark down parameters used by the storekeeper (expressed as a percentage of the actual value of the item).

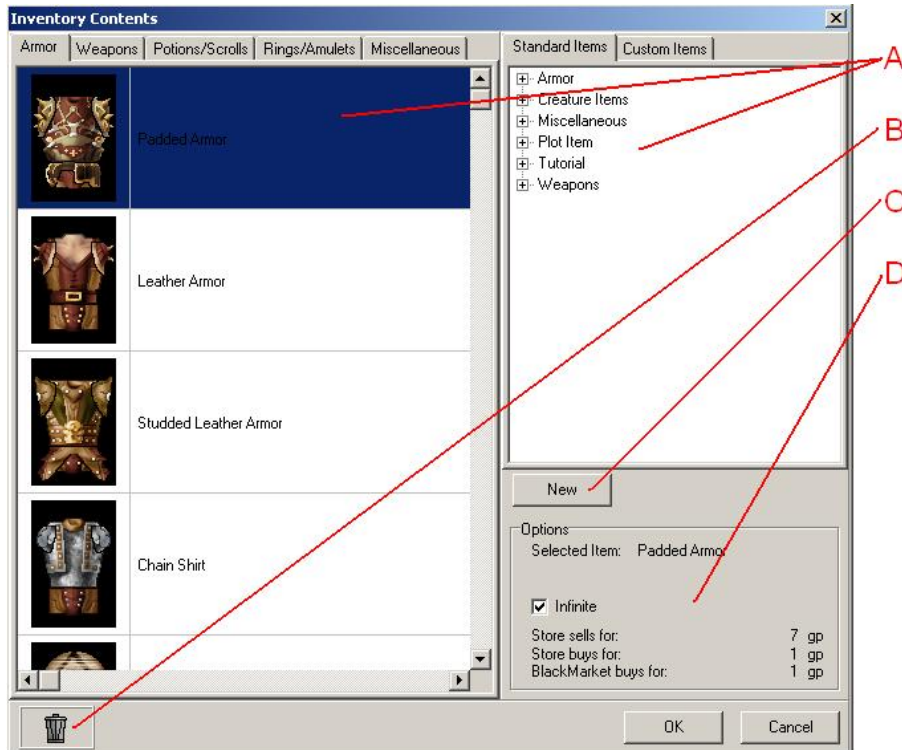
Stolen Goods: Items can be flagged as being stolen in their general properties. (See Sub-Section 7.3.1.) A storekeeper may not want to buy items that were acquired in an illegal fashion or, if they do, they may want to buy them at a lower price considering the risks involved in trading stolen goods. These parameters may be adjusted here.

Restrictions: The restrictions that may be adjusted here are related to the value of the items that can be sold to the storekeeper. You can either determine that a merchant will not buy items for a price higher than a specific amount or simply give a gold limit to the merchant (or do both).

If you put a gold limit to the merchant, note that this limit will be “interactive” in the game. The limit that you create here will be the actual amount of gold that the store has when the module starts, i.e. its treasury. If players buy items from the store, all gold given to make the trade will be added to store's treasury. When players sell items, the gold given in exchange will be reduced from the store's treasury. When there is not enough gold left in the treasury, the storekeeper will simply no longer buy any items.

Inventory: By clicking on the “Inventory...” button, you can determine what the store sells, and in what quantity it has them. See the inventory contents screen shown below.

Image 14.3.1b – Merchant Properties Screen – Inventory Contents

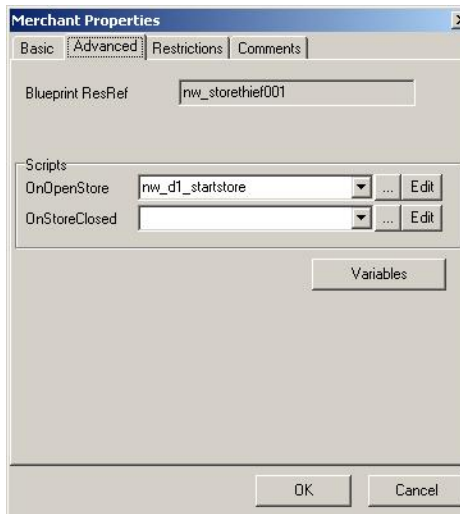


- A** You can add items offered for sale by choosing them from the item palettes. Double-click on an item in the palette to see it appear in the inventory contents of your store.
- B** Drag and drop an item in the inventory contents of your store to the trash can if you do not want the store to sell this item.
- C** By clicking on the “New” button, you will launch the Item Creation Wizard that will add custom items in the palette. See Section 7.1 for more details on the Item Wizard.
- D** There is one option that you can adjust here, which is to determine if you want the merchant to be able to sell an infinite number of the specific item that is selected or if you want this item to be unique in the inventory (i.e. once it is sold it will no longer appear as an item available for sale for this store). If you want the store to be able to sell, as an example, only 3 units of a specific item, then you will need to add this item three times to the inventory contents and make them all unique by disabling the “Infinite” property.

14.3.2 The Advanced Properties of a Merchant Object

The advanced properties of a merchant object are related to the scripts that are involved in the operation of the store.

Image 14.3.2 – Merchant Properties Screen – Advanced Tab



Blueprint ResRef: The blueprint ResRef is the unique reference identifying the merchant blueprint from which this merchant was created. This unique reference must be used when you want to create a store through a script or if you want to update all merchants already placed in your module in accordance with a modification you have made to the blueprint that was used to create them (this can be done by right-clicking on the blueprint and selecting the “Update Instances” in the dropdown menu).

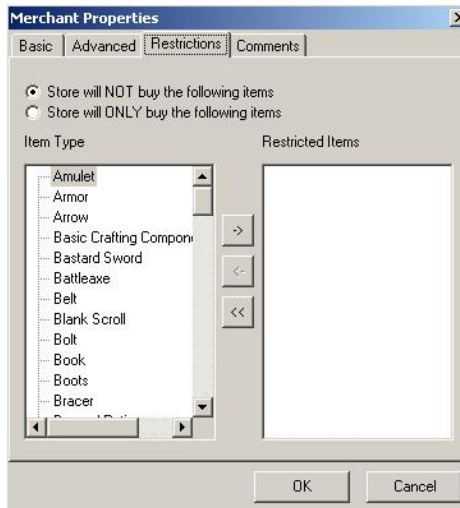
Scripts: These script events, as their names imply, will fire whenever the store opens (i.e. when a player asks the shopkeeper to show his wares) and when the store closes. The default script in the OnOpenStore event is empty and

does nothing. All the appraisal checks and restrictions are hard coded in the game and you cannot modify them.

Variables: The “Variables” assigned to a merchant object may be used in a script, but otherwise they do not have any effect.

14.3.3 The Restrictions of a Merchant Object

Image 14.3.3 – Merchant Properties Screen – We have seen in the basic properties of a merchant object that restrictions can be placed on the amount of gold that can be involved in a trade made through a store. The restrictions that can be customized under the restrictions tab are not related to pricing but rather to the type of items that a storekeeper agrees to buy.



The restrictions that can be customized under the restrictions tab are not related to pricing but rather to the type of items that a storekeeper agrees to buy.

A store operated by a guild of wizards may not have any interest at all in armor and arrows. You may therefore place here some categories of items on the "black list" of a store or, if this black list would be too long, you may identify which specific categories of items the storekeeper is interested in.

14.3.4 The Comments on a Merchant Object

The "Comments" tab on a merchant will provide you with a large text window in which you can write anything you like. You will notice merchant objects created from a standard merchant blueprint will already have a short comment, which will identify at which stage of the original campaign this merchant blueprint was used.

14.4 Designing Tips

Do not put too many restrictions on your merchant objects: Realism is one thing, but do not forget that your first goal is to build a module that is fun to play. Of course, it does not make any sense for a common innkeeper to be able to buy magical gear worth 10,000+ gold pieces. If the innkeeper had such a fortune in a medieval fantasy setting, he would not be an innkeeper. But if the innkeeper is the only merchant in the small town near a dungeon where a lot of treasure can be found, where will the players be able to sell the stuff they find? Forcing players to abandon treasures because they cannot find anyone who will buy it is a major fun breaker.

CHAPTER 15 Henchmen

Henchmen are creatures that can be hired by players to serve by their side. They might be merely a hireling or they might follow the players for another purpose. Their role in your module is for you to decide.

Note that the purpose of this Chapter is to provide you with the basics of henchmen creation. There are many tutorials and custom scripts available out there that deal with henchmen, each having their own approach towards henchmen. We therefore do not pretend that the method explained below is the only one, but we are of the opinion that it is the easiest one to incorporate in a module and to eventually customize when more advanced notions are integrated in a henchman's behaviour.

15.1 How to turn a Creature into a Henchman

The first step in creating a henchman is to create the creature that will be acting as such. Please refer to Chapter 4 for details on how to create a creature.

Now that you have created the creature, how do you make it behave like a henchman? The behaviour of any creature is handled by the scripting events that are assigned to it. Fortunately, default script events exist for creatures that are meant to be henchmen, and you may follow these easy steps to assign these default script events to your creature:

- 1) Open the creature's properties and click on the "Scripts" tab.
- 2) Click on the "Load Script Set" located at the bottom of the creature properties screen.
- 3) In the window that opens, select the file called "set_xp1_henchmen.ini" and click the "Open" button.

All the script events of your creature will then be changed for the default ones that handle henchman behaviour.

15.2 The Conversation File of a Henchman

Now that you have turned a creature into a potential henchman, you will need to determine how and when the henchman will accompany a player character. The easiest way to handle this, which is also the method used by Bioware in the original campaign, is by a conversation file that will be engaged between the player character and the henchman. Please refer to Chapter 5 for more details on how to create a conversation file for a creature.

It is important for the conversation file of a henchman creature to have at least three branches, namely:

- i) The node that is spoken when the Henchman has not yet joined the party.
- ii) The node that is spoken when the conversation is engaged by the henchman's master.

- iii) In multiplayer modules, the node that is spoken when the conversation is engaged by another player that is not the henchman's master.

You might also want to create more branches in addition to these absolutely essential conversation branches, especially those that will deal with the death system in your module (i.e. the node that is spoken when the henchman dies and come back to life) and those that will deal with the conversations about the quests that are being entrusted to player characters.

15.2.1 Conversation node spoken when the Henchman has not yet been hired

This conversation node attached to the root of the conversation file will be used to determine the conditions under which the henchman will follow the player character he is speaking to.

You must first make sure that this node will be spoken only when the henchman has no master. In order to do this, place the following script in the "Text Appears When..." property of this node:

```
int StartingConditional()
{
    object oMaster = GetMaster(OBJECT_SELF);
    if(GetIsObjectValid(oMaster)) return FALSE;
    return TRUE;
}
```

Next write the conversation that will eventually lead to the henchman joining the player. You can design this conversation branch as you wish: you can have the player barter to hire the henchman, or you can have the henchman graciously explaining why he would like to help the players in their adventure. But in the end, on the conversation node where the henchman confirms that he/she agrees to join the player, you will have to add the following script on the "Action Taken" property of this node:

```
void main()
{
    object oSpeaker = GetPCSpeaker();
    object oHenchman = OBJECT_SELF;
    AddHenchman(oSpeaker, oHenchman);
}
```

15.2.2 Conversation node spoken between a hired Henchman and his Master

This conversation node attached to the root of the conversation is usually used to deal with the instructions that players give to their henchman.

You must first make sure that this node will be spoken only when the henchman's master speaks to him. In order to do this, place the following script in the "Text Appears When..." property of this node:

```

int StartingConditional()
{
object oSpeaker = GetPCSpeaker();
object oMaster = GetMaster(OBJECT_SELF);
if(oSpeaker != oMaster) return FALSE;
return TRUE;
}

```

Then determine what options you want to give to players in the instructions given to the henchman. The conversation branch is for you to write as you wish, but there are common elements in most custom made henchman conversations.

Below is a list of scripts that can be used in those common instructions nodes. These scripts must be placed on the "Action Taken" property of the node wherein the player is giving the relevant instruction to the henchman.

- i) Player gives instructions to no longer follow him:

```

void main()
{
object oSpeaker = GetPCSpeaker();
ClearAllActions();
RemoveHenchman(oSpeaker);
}

```

- ii) Player gives instructions to defend him/her and not to attack before the player engages into battle:

```

#include "x0_i0_assoc"
void main()
{
SetAssociateState(NW_ASC_MODE_DEFEND_MASTER);
}

```

- iii) Player gives instructions to attack as soon as the henchman sees hostile creatures:

```
#include "x0_i0_assoc"
void main()
{
SetAssociateState(NW_ASC_MODE_DEFEND_MASTER, FALSE);
}
```

- iv) Player gives instructions to change the distance at which the henchman is following him/her:

Use following script when the player asks the henchman to stay close:

```
#include "NW_I0_GENERIC"
void main()
{
SetAssociateState(NW_ASC_DISTANCE_2_METERS, TRUE);
SetAssociateState(NW_ASC_DISTANCE_4_METERS, FALSE);
SetAssociateState(NW_ASC_DISTANCE_6_METERS, FALSE);
}
```

On nodes where the player asks the henchman to stay at medium or long range, use the above script but make the appropriate state to "TRUE" and those that are not applicable "FALSE".

- v) Player gives instructions to help when he/she comes across locked chests or doors:

```
#include "NW_I0_GENERIC"
void main()
{
SetAssociateState(NW_ASC_RETRY_OPEN_LOCKS, TRUE);
}
```

Use the same script when the player gives instructions not to help, but the value "TRUE" must be changed to "FALSE".

- vi) Player gives instructions to the henchman to be stealthy:

```

void main()
{
ClearAllActions();
SetLocalInt(OBJECT_SELF, "X2_HENCH_STEALTH_MODE", 1);
DelayCommand(1.0, SetActionMode(OBJECT_SELF, ACTION_MODE_STEALTH,
TRUE));
}

```

If the instructions are to be stealthy, but only until there is a combat, use the same script template but the value of "1" given on the line that starts with the SetLocalInt command must be changed to "2". If the instructions are to no longer be stealthy, then the value must be changed to "0".

vii) Player gives instructions to help disarm traps

The following script must be used when the player asks the henchman to help in that regard:

```

#include "NW_I0_GENERIC"
void main()
{
SetAssociateState(NW_ASC_DISARM_TRAPS, TRUE);
}

```

Use the same script when the player gives instructions not to help, but the value "TRUE" must be changed to "FALSE".

There are many more actions you may want to give to your henchman creatures, like casting spells and using class related abilities. The best way to find templates for scripts to accomplish these goals is to open the conversation files of the henchmen used in the official campaign and to look at the conversation, find the node you are looking for and look at the scripts that Bioware has used. These conversation files may be accessed by:

- 1) Launching the Conversation Editor;
- 2) Clicking on the "Open an Existing File" button in the top button bar; and
- 3) When the "Select Resource" screen opens, tick the "All Resources" option at the bottom left corner of this screen.

Conversation files used in the official campaign will then be added to the list of available resources. Look for the conversation files that start with "xp2_hen_" and open the one that is of interest to you.

Copy and paste to your henchman conversation file all the nodes that you find interesting, and edit them as necessary.

15.2.3 Conversation node spoken between a hired Henchman and a player that is not his Master

This conversation node that must be attached to the root of the conversation is relevant only if your module is intended to be playable in multiplayer mode. The purpose of this node is to make sure that other players cannot either steal another player's henchman or to change the instructions given to a henchman that he does not control.

In order to make sure that this node is spoken only when a player that is not the master of the henchman engages in conversation, place the following script in the "Text Appears When..." property of this node:

```
int StartingConditional()
{
    object oSpeaker = GetPCSpeaker();
    object oMaster = GetMaster(OBJECT_SELF);
    if(!GetIsObjectValid(oMaster)) return FALSE;
    if(oSpeaker == oMaster) return FALSE;
    return TRUE;
}
```

You might not want to make a very detailed conversation branch for this purpose. You can simply leave this node as a one-liner by adding a text like "I don't wish to talk to you." or something equivalent.

15.3 Designing Tips

Henchmen are fun, but they should not be the hero: Henchmen can play an important part in a NWN module. They give the builder the opportunity to deepen the story and provide more opportunity to have larger and more strategic battles. There is one thing that must not be forgotten however; the player is the hero, the henchman or henchmen are only there to help him. The henchman should not be so strong that he ends up doing the job all by himself.

Give reasonable gear to the henchman consistent with his/her background and the difficulty level of the module: The henchman should also have inventory that fits with the story and the setting. If you meet a low level henchman in the inn of a small village, there is no reason why he would have an inventory full of magic items. Having magic items implies that this henchman has done a lot of adventuring before, which does not seem to be logical.

Give a purpose to the henchman, explain how he/she fits in the story: As an example, we often see in modules henchmen (sometimes high level characters) standing in an inn near a dungeon, waiting to be hired. Why would an almost epic hero be there, waiting for someone to hire him? The conversation file of the henchman needs to address this issue in order for your story to remain logical.

15.4 How to...

15.4.1 ...allow players to have more than one henchman

This can be done quite easily. Add the following line to the OnModuleLoad script of your module's properties:

```
SetMaxHenchmen(3);
```

As you have probably figured out, the number between brackets is the maximum number of henchmen that is allowed per player.

15.4.2 ...to bring a henchman back to life only if a resurrection / raise dead spell is cast on him/her

- 1) Change the OnDeath script of your henchman creature (under the "Scripts" tab of its properties) for the following one: x2_hen_death
- 2) Change the OnSpellCastAt script of your henchman creature (under the "Scripts" tab of its properties) for the following one: x2_hen_spell

CHAPTER 16 x2_inc_switches

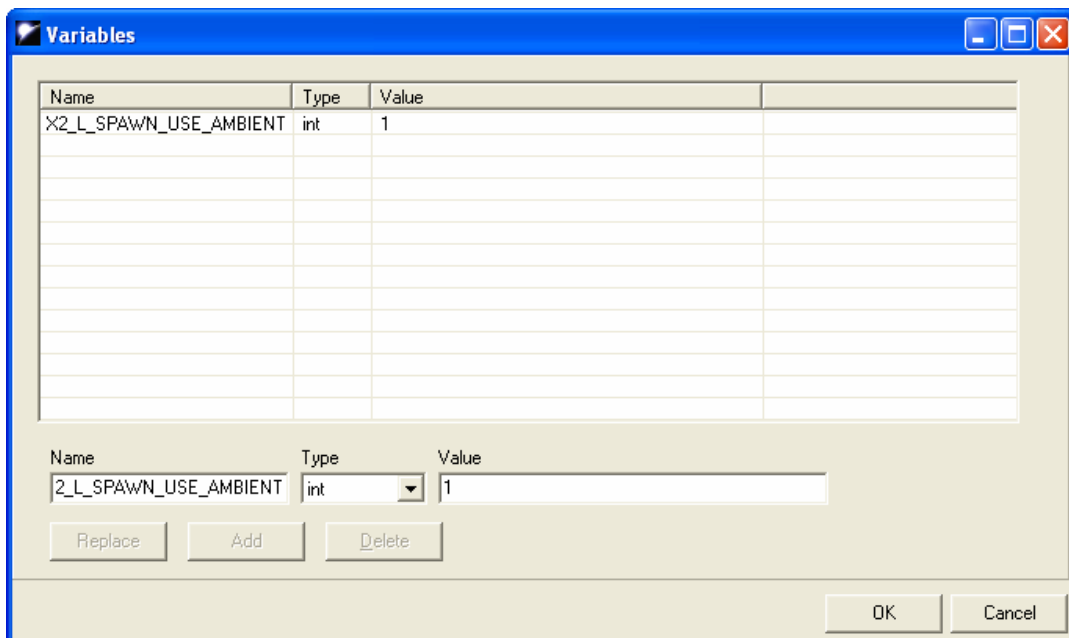
16.1 Interface for switching subsystem functionality

The default scripts introduce with the Hordes of the Underdark expansion added the ability to control many aspect of the game using a simple variable setting system Following are the instructions on how to use this system. In the sections below any reference to a value of false means the integer has a numeric value of 0 and any reference to a value of true means the integer has a numeric value of 1

This file provides a basic interface for switching different Hordes of the Underdark subsystems on/offhand allows centralized access to certain "expert" functionality like overriding AI or Spellscripts. Changing any of these switches from their default position is considered unsupported and done at your own risk - please do NOT send any bug reports about problems caused by these switches.

*Copyright (c) 2003 BioWare Corp. * Created By: Georg Zoeller * On: 2003-07-16*

Example variable screen



The screenshot shows a window titled "Variables" with a table and input fields. The table has three columns: Name, Type, and Value. The first row contains "X2_L_SPAWN_USE_AMBIENT", "int", and "1". Below the table are input fields for Name, Type, and Value, with "2_L_SPAWN_USE_AMBIENT", "int", and "1" respectively. There are buttons for "Replace", "Add", "Delete", "OK", and "Cancel".

Name	Type	Value
X2_L_SPAWN_USE_AMBIENT	int	1

Name: 2_L_SPAWN_USE_AMBIENT Type: int Value: 1

Buttons: Replace, Add, Delete, OK, Cancel

16.1.1 Module Switches

Force Use Magic Device Skillchecks, Default = FALSE(0) except for GAME_DIFFICULTY_CORE_RULES+ If switched to TRUE, a rogue has to succeed in a UMD check against DC 25+SpellLevelin order to use a scroll. See x2_pc_umdcheck.nss for details

X2_SWITCH_ENABLE_UMD_SCROLLS = 1

Toggle on/off the Item Creation Feats, Default = ODisable the Item Creation Feats that come with Hordes of the Underdark for themodule.

X2_SWITCH_DISABLE_ITEMCREATION_FEATS=1

Toggle Area of Effect Spell behaviourIf set to TRUE, AOE Spells will hurt NPC'S that are neutral to the caster if they arecaught in the effect.

X0_G_ALLOWSPELLSTOHURT = 1

For balancing reasons the crafting system will create 50 charges on a new wand instead it will create 10 + casterlevel charges. if you want to be "hard core rules compliant"50 charges, enable this switch

X2_SWITCH_ENABLE_50_WAND_CHARGES = 1

Some epic spells, namely Hellball, do damage to the caster. We found this too confusingin testing, so it was disabled. You can reactivate using this flag

X2_SWITCH_EPIC_SPELLS_HURT_CASTER = 1

Deathless master touch is not supposed to affect creatures of size > largebut we do not check this condition by default to balance the fact that the slaincreature is not raised under the command of the pale master.by setting this switch to TRUE, the ability will no longer effect creatures ofhuge+ size.

X2_SWITCH_SPELL_CORERULE_DMTOUCH = 1

By default, all characters can use the various poisons that can be found to poison their weapons ifthey win a Dex check. Activating this flag will restrict the use of poison to chars with the UsePoisonfeat only

X2_SWITCH_RESTRICT_USE_POISON_FEAT

Multiple Henchmen: By default, henchmen will never damage each other with AoE spells.By activating this switch, henchmen will be able to damage each other with AoE spellsand potentially go on each other's throats.Warning: Activating this switch has the potential of introducing game breaking bugs. Do not use on the official SoU campaign. Use at your own risk. Really, its dangerous!

X2_SWITCH_MULTI_HENCH_AOE_MADNESS

Spell Targeting: Pre Hordes of the underdark, in hardcore mode, creatures would not hurt each other with their AOE spells if they were no PCs. Setting this switch to true, will activate the correct behaviour. Activating this on older modules can break things, unless you know what you are doing!

X2_SWITCH_ENABLE_NPC_AOE_HURT_ALLIES

If set to TRUE, the Babilith Ruin Armor ability is going to actually destroy the armor it hits. Would be very annoying for players...

X2_SWITCH_BEBILITH_HARDCORE_RUIN_ARMOR

Setting this switch to TRUE will make the Glyph of warding symbol disappear after 6 seconds, but the glyph will stay active....

X2_SWITCH_GLYPH_OF_WARDING_INVISIBLE

Setting this switch to TRUE will enable the allow NPCs running between waypoints using the WalkWaypoints function to cross areas, like they did in the original NWN. This was changed in 1.30 to use only waypoints in one area.

X2_SWITCH_CROSSAREA_WALKWAYPOINTS

Setting this switch to TRUE will disable the glow of a newly found secret door used in some locations in XP2

X2_SWITCH_DISABLE_SECRET_DOOR_FLASH

Setting this switch to TRUE will disable execution of tag based scripts that are enabled by default when using the standard module events (x2_mod_def_*)

X2_SWITCH_ENABLE_TAGBASED_SCRIPTS

Setting this switch to TRUE will enable the XP2 Wandering Monster System for this module (if you are using the default rest script and you have setup the correct variables for each area)

X2_SWITCH_ENABLE_XP2_RESYSTEM

If this variable is set, the AI will not use Dispel Magic against harmful AOE spells.

X2_L_AI_NO_AOE_DISPEL

Setting this variable to TRUE on the module will disable the call to the random loot generation in most creatures' OnSpawn script.

X2_L_NOTREASURE

16.1.2 Miscellaneous

X2_S_UD_SPELLSCRIPT

Variable that holds the wandering monster 2da filename

X2_WM_2DA_NAME

This variable allows to specify a % for NOT using dispel magic against AOE's instead fleeing

X2_L_AI_AOE_DISPEL_CHANCE

Setting this variable to TRUE will cause the Expertise/Improved Expertisemodes to be disabled whenever a player is casting a spell.

X2_L_STOP_EXPERTISE_ABUSE

16.1.3 Creatures

Define a replacement script for DetermineCombatRound See x2_ai_demo for details Call this at end of your custom override AI script set via CREATURE_VAR_CUSTOM_AISCRIP See x2_ai_demo for details. See x2_ai_demo for more details

X2_SPECIAL_COMBAT_AI_SCRIPT

Setting this variable on a spellcaster creature will make its spell use a bit more random, but their spell selection may not always be appropriate to the situation anymore.

X2_SPELL_RANDOM

Set to 1 to make the creature activate stealth mode after spawn

X2_L_SPAWN_USE_STEALTH

Set to 1 to make the creature activate detect mode after spawn

X2_L_SPAWN_USE_SEARCH

Set to 1 to make the creature play mobile ambient animations after spawn

X2_L_SPAWN_USE_AMBIENT

Set to 1 to make the creature play immobile ambient animations after spawn

X2_L_SPAWN_USE_AMBIENT_IMMOBILE

Set to 1 to make the creature immune to dispel magic (used for statues)

X1_L_IMMUNE_TO_DISPEL

Set this variable to 1 on a creature to make it walk through other creatures

X2_L_IS_INCORPOREAL

Set this variable to 1 - 6 to override the number of attacks a creature has based on its BAB

X2_L_NUMBER_OF_ATTACKS

The value of this variable (int) is added to the chance that a creature will use magic in combat. Set to 100 for always, 0 for never

X2_L_BEH_MAGIC

The higher value of this variable, the higher the chance that the creature will use offensive abilities in combat. Set to 0 to make them flee.

X2_L_BEH_OFFENSE

The higher value of this variable, the higher the chance that the creature will aid friendly creatures in combat. Not that helping usually degrades the overall difficulty of an encounter, but makes it more interesting.

X2_L_BEH_COMPASSION

This allows you to script items that enhance a palemaster's summoned creatures. You need to put the name of a script into this variable that will be run on any creature called by the pale master's summon undead ability. You can use this script to add effects to the creature. You can use the OnEquip/OnUnEquip event hooks set this variable.

X2_S_PM_SPECIAL_ITEM

16.1.4 Doors

Set to 1 to prevent knock from working with feedback. Set to 2 to prevent knock from working without feedback

X2_FLAG_DOOR_RESIST_KNOCK

16.1.5 Waypoints

By setting this variable to 1 on a waypoint, a creature using this waypoint as part of its Walk Way Points routine will assume the facing of the waypoint upon reaching it.

X2_L_WAYPOINT_SETFACING

16.1.6 Items

16.1.7 Spellscripts

Allows the module creator to specify a script that will be run before any spellscript is run You can call SetModuleOverrideSpellscript() at the end of the script specified by sScriptName. If you call this function it will prevent the original spellscript (and all craft item code) from being executed. If you do not add this line, the original spellscript and/or crafting code will run in addition to your script

16.2 Doing it in Hex code

You can modify all spawn conditions of a creature by putting an integer variable called "NW_GENERIC_MASTER" on the creature and put the correct value calculated from the values below:

```
int NW_FLAG_SPECIAL_CONVERSATION          = 0x00000001;
int NW_FLAG_SHOUT_ATTACK_MY_TARGET        = 0x00000002;
int NW_FLAG_STEALTH                        = 0x00000004;
int NW_FLAG_SEARCH                         = 0x00000008;
int NW_FLAG_SET_WARNINGS                   = 0x00000010;
int NW_FLAG_ESCAPE_RETURN                  = 0x00000020; //Failed
int NW_FLAG_ESCAPE_LEAVE                   = 0x00000040;
int NW_FLAG_TELEPORT_RETURN                 = 0x00000080; //Failed
int NW_FLAG_TELEPORT_LEAVE                 = 0x00000100;
int NW_FLAG_PERCIEVE_EVENT                 = 0x00000200;
int NW_FLAG_ATTACK_EVENT                   = 0x00000400;
int NW_FLAG_DAMAGED_EVENT                  = 0x00000800;
int NW_FLAG_SPELL_CAST_AT_EVENT            = 0x00001000;
int NW_FLAG_DISTURBED_EVENT                = 0x00002000;
int NW_FLAG_END_COMBAT_ROUND_EVENT         = 0x00004000;
int NW_FLAG_ON_DIALOGUE_EVENT              = 0x00008000;
int NW_FLAG_RESTED_EVENT                   = 0x00010000;
int NW_FLAG_DEATH_EVENT                    = 0x00020000;
int NW_FLAG_SPECIAL_COMBAT_CONVERSATION    = 0x00040000;
int NW_FLAG_AMBIENT_ANIMATIONS             = 0x00080000;
int NW_FLAG_HEARTBEAT_EVENT                = 0x00100000;
int NW_FLAG_IMMOBILE_AMBIENT_ANIMATIONS    = 0x00200000;
int NW_FLAG_DAY_NIGHT_POSTING              = 0x00400000;
```



```

int NW_FLAG_AMBIENT_ANIMATIONS_AVIAN          = 0x00800000;
int NW_FLAG_APPEAR_SPAWN_IN_ANIMATION        = 0x01000000;
int NW_FLAG_SLEEPING_AT_NIGHT                = 0x02000000;
int NW_FLAG_FAST_BUFF_ENEMY                  = 0x04000000;

```

So if you want a creature to be in search and stealth mode on spawning and to use ambient animations, you can set "NW_GENERIC_MASTER" to

```

0x00000004 +
0x00000008 +
0x00080000 =

```

```

0x0008000C = 524300

```

```

const string sCombatCondVarname = "X0_COMBAT_CONDITION";

const int X0_COMBAT_FLAG_RANGED          = 0x00000001;
const int X0_COMBAT_FLAG_DEFENSIVE      = 0x00000002;
const int X0_COMBAT_FLAG_COWARDLY     = 0x00000004;
const int X0_COMBAT_FLAG_AMBUSER      = 0x00000008;

```

You need to put the value as decimal (the bold value) on the creature. You can use windows calculator (switch to scientific mode) to calculate the hex numbers together and then convert them to decimal mode.

NW_FLAG_SPECIAL_CONVERSATION = 1
NW_FLAG_SHOUT_ATTACK_MY_TARGET = 2
NW_FLAG_STEALTH = 4
NW_FLAG_SEARCH = 8
NW_FLAG_SET_WARNINGS = 16
NW_FLAG_ESCAPE_RETURN = 32
NW_FLAG_ESCAPE_LEAVE = 64
NW_FLAG_TELEPORT_RETURN = 128
NW_FLAG_TELEPORT_LEAVE = 256
NW_FLAG_PERCIEVE_EVENT = 512
NW_FLAG_ATTACK_EVENT = 1024
NW_FLAG_DAMAGED_EVENT = 2048
NW_FLAG_SPELL_CAST_AT_EVENT = 4096
NW_FLAG_DISTURBED_EVENT = 8192
NW_FLAG_END_COMBAT_ROUND_EVENT = 16384
NW_FLAG_ON_DIALOGUE_EVENT = 32768
NW_FLAG_RESTED_EVENT = 65536
NW_FLAG_DEATH_EVENT = 131072
NW_FLAG_SPECIAL_COMBAT_CONVERSATION = 262144
NW_FLAG_AMBIENT_ANIMATIONS = 524288
NW_FLAG_HEARTBEAT_EVENT = 1048576
NW_FLAG_IMMOBILE_AMBIENT_ANIMATIONS = 2097152
NW_FLAG_DAY_NIGHT_POSTING = 4194304
NW_FLAG_AMBIENT_ANIMATIONS_AVIAN = 8388608
NW_FLAG_APPEAR_SPAWN_IN_ANIMATION = 16777216
NW_FLAG_SLEEPING_AT_NIGHT = 33554432

NW_FLAG_FAST_BUFF_ENEMY = 67108864

16.2.1 For those who are clueless like I was...

open your windows calculator in scientific mode and check the HEX box.

Now refer to the onspawn trigger/event you want to use.

For example, you want to turn on the ambient animations.

You see the HEX number for ambient animations is

0x00080000.

Type into the calculator 80000.

Now click the DEC check box.

The number it gives you is what you set NW_GENERIC_MASTER to.

To turn on more than one, IN HEX mode, enter the HEX NUMBERS and add them normally. when you are finished click the DEC box.

Again, there is your setting for NW_GENERIC_MASTER

CHAPTER 17 XP2 (HotU) Rest / Wandering Monster System

17.1 The basics

This feature introduced with the Hordes of the Underdark expansion allows the builder to setup a pseudo wandering monster system. Essentially the way it works is every time the player enters a new area the engine checks for a variable set on the area the sets a 2da table (wandering monster table) for that area. When the player rests it runs some random checks and spawns monsters according to the table for that area. If no custom table is set then the default table is used. Note that the default table is from HotU and will probably not be appropriate for non Underdark settings. More information can be found in the script `x2_inc_restsys`. Below is a brief rundown of how to setup this system in your modules.

- a) Make sure you are using the default `OnModuleLoad` script `x2_mod_def_load`.
- b) Make sure you are using the default `OnPlayerRest` script `x2_mod_def_rest`.
- c) `x2_inc_restsys`, the rest script actually has an explanation of which variables you want to set on an area
- d) Set the integer variable `MODULE_SWITCH_USE_XP2_RESYSTEM=1` on the module
- e) You need to modify your `des_restsys.2da`, the 2da file which has the "template" definitions for wandering monsters
- f) `x2_inc_restsys.nss` IS the official rest system. Its commented so reading through it should give you some ideas.
- g) Open an area `Edit>Properties>Advanced>Variables` add this variable `X2_WM_ENCOUNTERTABLE` string Forest and you will get deer or dogs. You can also set this string to the name of your own custom table for that area.

You don't need the custom 2da for it to work that's optional without it the game will default to the stock one which is fairly basic.

Note: You can call `WMSetAreaProbability` later to change the probability of having an encounter .

In the areas where you want the wandering monsters to appear set the `X2_WM_ENCOUNTERTABLE` string variable to the name of the table to be used. The creatures will spawn in at any object with the tag `x2_restsys_spawn` or any door

Comment: Need to find out about the "door System" mentioned in the scripts.

Note: this corresponds to the "TableName" column from `des_restsystem.2da`

If you want to add safe place in areas you can place triggers that will set the X2_WM_DISABLED variable on the area to TRUE when the PC enters the trigger and to false when the PC leaves the trigger. See the script samples below.

For the OnEnter Event to turn off the wandering monster system.

```
//Put this script OnEnter of a trigger to turn the wandering monster system OFF for that area
void main()
{
object oPC = GetEnteringObject();
object oArea = GetArea(OBJECT_SELF);
if (!GetIsPC(oPC)) return;
SetLocalInt(oArea, "X2_WM_DISABLED", 1);
}
```

For the OnExit event of a trigger to turn the wandering monster system back on.

```
//Put this script OnExit of a trigger to turn the wandering monster system ON for that area
void main()
{
object oPC = GetExitingObject ();
object oArea = GetArea(OBJECT_SELF);
if (!GetIsPC(oPC)) return;
SetLocalInt(oArea, "X2_WM_DISABLED", 0);
}
```

17.2 Making your own wandering monster table (.2da)

To do this you will need to extract the file des_restsystem.2da from the game resources. You can do this with the [Modified NWN Explorer](#) by roboius. You can also download all the games 2da files from the BioWare site [here](#). Once you have the des_restsystem.2da you can open it using any plain text editor like windows notepad. From here you will want to look over the table carefully before making any changes. Below is a list of the columns in the table and what each one does.

Note: There are 3 day and 3 night ResRef columns this allows for anyone one of the three possible day/night groups to spawn in each encounter.

TableName: This is the name of the row in the table and is used whenever the player attempts to rest.

Comment: you can add a brief description of this row here

ListenCheckDC: affect the chance of the player making a successful listen check to detect the presence of a wandering monster

FeedBackStrRefSuccess: calls a line from the TLK table to let the player know if they have heard something.

FeedBackStrRefFail: Message if the player fails their listen check.

DayBaseProbability: percent chance of a random encounter when resting during the day time.

NightBaseProbability: percent chance of a random encounter when resting during the night time.

DAY_ResRef1: this is where the resrefs used for the first group of creatures that can be spawned for a day time wandering monster encounter are listed. If more than one creature is to be spawned separate them with a "," comma.

DAY_Prob1: the chance that the creature from DAY_ResRef1 will be spawned.

DAY_ResRef2: the second possible group.

DAY_Prob2: the chance that the second group will be spawned

DAY_ResRef3 the third possible group.

Note that there is no DAY_Prob3. This is the default group and will be spawned if 1 or 2 do not get called.

NIGHT_ResRef1: same as the day column but for night time encounters

NIGHT_Prob1: same as the day column but for night time encounters

NIGHT_ResRef2: same as the day column but for night time encounters

NIGHT_Prob2: same as the day column but for night time encounters

NIGHT_ResRef3: same as the day column but for night time encounters

CHAPTER 18 Using the SoU Treasure System

This information is taken from the comments in the X0_I0_TREASURE script. X0_I0_TREASURE is the brains behind the SoU treasure system. Read carefully the comments here and you will see that this script is basically a tool to create your own random treasures without having to write any scripts at all.

18.1 Main Concept:

The module builder creates some base containers and fills them with instances of the treasure items that they want to appear randomly. The actual treasure containers then randomly pick from the contents of these base containers to generate their own contents.

This means that there is a built in tool for making your own random treasure spawns for containers (and other things but we will get to that latter). The next section explains just how simple it is to setup your own custom container spawns.

18.2 Container Setup:

Create four module-wide base containers (for low, medium, high, and unique treasure items) and place them anywhere in your module. Make sure that they are in locations inaccessible to the players, though! Give these module-wide base containers these tags (or just use the blueprints from the palette):

- X0_MOD_TREASURE_LOW - low-level treasure
- X0_MOD_TREASURE_MED - medium-level treasure
- X0_MOD_TREASURE_HIGH - high-level treasure
- X0_MOD_TREASURE_UNIQ - unique treasure items

Fill the instances of these base containers with appropriate treasure for your module. For any areas where you want finer control over the treasure, simply create additional base containers (you don't need all four -- any one that you skip will fall back to the module-wide version) and place them in a part of the area not accessible to the PC. Give them these tags (same as above just without the "MOD"):

- X0_TREASURE_LOW - low-level treasure
- X0_TREASURE_MED - medium-level treasure
- X0_TREASURE_HIGH - high-level treasure
- X0_TREASURE_UNIQ - unique treasure items

For any treasure container, use one of the following scripts as BOTH the OnOpen and OnDeath event handler:

Any Treasure: x0_o2_any{low,med,high,uniq}

Books (book, scroll): x0_o2_book{low,med,high,uniq}

Potions: x0_o2_potn{low,med,high,uniq}

Armor (armor, shield, helm, boots, etc): x0_o2_arm{low,med,high,uniq}

Weapon: x0_o2_weap{low,med,high,uniq}

Gold: x0_o2_gold{low,med,high,uniq}

The container will then randomly select items of the corresponding type from the area or module containers of the specified level to spawn in.

18.3 MONSTER/NPC TREASURE

If you would like to have special monster treasure (monster treasure defaults to low treasure otherwise), you can also add chests that will let you customize their loot generation.

To use monster treasure, use the default OnSpawn script nw_c2_default9 or modify the OnSpawn script as follows:

- Replace #include "nw_o2_coninclude" with #include "x0_i0_treasure"
- Replace GenerateNPCTreasure(); with CTG_GenerateNPCTreasure();
- If you prefer to generate monster treasure from the general chests, you can also add a parameter to CTG_GenerateNPCTreasure(); to specify whether the treasure generated should be low/medium/high/unique. See the comments to that function for details.

18.3.1 Tag based Monster Treasure

Set the tag of the chest to: X0_TREASURE_MONSTER_<monster tag>

Treasure found on monsters with the given tag will get their loot from that chest. This will strip off any trailing digits, so for instance,

- NW_KOBOLD1
- NW_KOBOLD2
- NW_KOBOLD005

Would all end up checking the chest X0_TREASURE_MONSTER_NW_KOBOLD.

18.3.2 Racial Type Based Monster Treasure

Set the tag of the chest to: X0_TREASURE_MONSTER_<racialtype>

Treasure found on monsters of the given racialtype will then use that chest when generating their loot. Examples:

- X0_TREASURE_MONSTER_ELF
- X0_TREASURE_MONSTER_UNDEAD

The spelling matches that used in the RACIALTYPE_* constants.

18.3.3 Generic Monster Treasure

To cover the monsters not using tag or racial type chests you can designate a generic chest by using the tag X0_TREASURE_MONSTER. This will generate treasure found on all monsters/NPCs in the module not covered by a tag or racial type chest.

DETAILS:

Each treasure container has a script that looks for the nearest object with the tag matching the type of treasure. When the treasure-generating scripts look for a container, they will take the nearest container in the local area, then fall back to the module-wide container. If no container can be found, they will fall back on the standard random treasure generation system. The treasure generation system will randomly select from 1-3 items from whatever items are in the nearest container. Each item has a 50/50 chance of being replaced by a random amount of coin based on the standard treasure generation system.

IMPORTANT NOTE ON PROBABILITY:

Each item in the chest has an equal probability of being selected. To control the probability of any given item appearing, simply place multiple copies of it in the container. Because of the stacking rules, there is a slight modification to this. Ammunition/throwing weapons will have a probability equal to the number of stacks in the chest, since the max stack size is the default stack size. Potions, scrolls, and gems, however, will be counted individually to determine the probability even if they are stacked, since their default stack size is 1. To control the size of the stack that actually gets created, you should create a blueprint that has the appropriate stack size. You can adjust the stack size of the instance you stuff into the base container to whatever you want afterwards; the blueprint's stack size will control how many are in the stack generated. There are scripts that will cause only specific types of items to be selected from the container of the specified level --

eg, only books/scrolls, or only weapons or armor, etc – useful for bookcases or for weapon racks, & so on. The unique treasure chest will only place each item in the chest once, will never place more than one item, and will never replace the item with coin.

Advantages of this system: - Complete control over which items can appear in a module or in an area. - Thematically-appropriate treasure can be placed in each in each area.

- Easy for module builders to set up.
- Trivial to add new items to the treasure distribution.
- Trivial to adjust probabilities of items appearing.

- Easy to have unique treasure items showing up randomly.

Disadvantages of this system:

- No level-specific treasure generation. This system is thus NOT appropriate for an area that is intended to scale to radically-different levels. Rather, it is appropriate for use where the levels of the characters are roughly known to the module designer and the treasure can be designed to match.
- No class-specific treasure generation planned for now. (However, I may add something to increase probability of certain items popping up, optionally. TBD.) For XP1, we feel this is a good tradeoff, but this should be kept in mind when deciding whether to use this system or the original treasure generation system in user-made modules.

CHAPTER 19 Other Stuff

19.1 Adding Crafting Materials to your campaign.

In order to allow your players to use their crafting skills, you need to make sure that crafting materials are available to them.

The easiest way to do that is to add the script `x2_o2_dead.nss` to the `OnDeath` event of destroyable placeables and doors, like it is used in the official HotU campaign. This will cause the placeable/door to drop some crafting materials if it is destroyed by a character skilled enough in crafting (craft armour or craft weapon 5+) to salvage the material. The default death scripts will take care of dropping creature related material (i.e. hides, feathers or metal for certain types of golems).

The materials that are dropped by this script are defined in `des_craft_drop.2da` where the row index in that file corresponds to the appearance type of the placeable/door/creature calling the script.

Tip:

PW Builders Note: You do not need to distribute a customized version of this file if you make any changes to it (i.e. via `hakpak`), as the script using the file is server side only (this way you can hide your drop tables from your players).

19.2 Using HAKs to patch your module after release.

This is a simple way to keep your module up to date with any fixes made after it is released without forcing your players to restart their games. This will even work with Persistent Worlds and can simplify the update process for all mod builders. Special thanks to Lance Botelle for writing this up and bringing it to our attention.

19.2.1 The Purpose:

To allow players to simply update the module they are playing with latest fixes without having to start the module again.

19.2.2 The Concept:

Use the `hak` facility to update code in a module that may require fixing.

Using a `hak` to update code is possible because anything in a `hak` takes priority over the code that is within a module.

19.2.3 Preparation:

The only real preparation for the builder is to have an "empty" hak (containing nothing apart from a readme.txt explaining what the hak is there for) that is separate from any other hak, which is associated with the module at build time, and that can easily be updated as patches are required.

19.2.4 Patching Procedure:

Should the builder have the need to change a script or a conversation, or even add items, then all they need to do, is have access to the "temp" folder within the modules directory while they make the alterations.

Inside this temp directory are all the files that *can* be added to the hak to make alterations to original mod. For example, if you alter a script (and compile it), *before* saving the module, you can go to the temp directory (alter view by latest files updated) and notice the NCS and NSS files have been updated when last compiled. In this case, it is the NCS (compiled script) that we need to add to the hak that will take priority over the current one in the module. This could just as easily been a DLG (conversation) that had just been altered in some way.

19.2.5 Considerations:

Sometimes, resolving the exact problem may not always be obvious, and the builder may have to use other scripts to help resolve the problem in the module. Here is an example of a problem I had with Soul Shaker that helps demonstrate the patching facility at work:

19.2.6 Module Problem Example:

A situation was reported where someone could get themselves locked inside a room without the key to get out.

PATCHING PLAN OF ACTION:

I had to find a way of doing the following:

- a) Check if the player was in this situation.
- b) Work out a way of adding the key to somewhere in the room if this was the case.

19.2.7 The Patch:

This problem also required the need for a new item (the key), so in the toolset I built a new key blueprint (that would fit the door) and grabbed that from the temp folder (it was a UTI file). Then I did the following: There was a trigger in the room and so I altered its Onenter script to check if the player fell in the "patch required" category (did they need the key!). If they did, then the new script now created the new key on a nearby humanoid corpse placeable, which I located in the script with GetNearestObjectByTag, which was in the room.

19.2.8 The Patch at Work:

So, I added the new UTI (key item) and NCS (trigger Onenter) files to the "unique" hak that I reserved for this purpose and uploaded it for people to use instead of the previously empty one. Now, when the module plays, it will refer to the scripts in the hak as priority over the "older" ones in the module and play out as explained above.

NB: I imagine there may be *some* situations where there is no easy way to "disguise" a patch at work. In other words, if this room in the example above had no access to any nearby objects with scripts, then there may be no other solution than to add the "patch" as an alteration to the players OnEnter script if they meet the "pachting" requirements and simply give them the item as they enter the game. Of course, if it is just a variable or other simple scripting process that needs fixing, then this should be very simple to do.

19.2.9 Later Patches:

Later patches simply continue to add to this hak as changes are made.

I recommend you keep altered scripts in the newer module (even if you resolve the problem by simply changing the module - like placing a key in the room in the example above) because your altered scripts will then still be available for those who are playing an older version of the module and still *require* the altered code.


19.2.10 Final Words:

I have only tried scripts, conversations and items, but it may be possible to add/do other things that I have not tried yet.

E.g. I have not yet looked for any way (or know if it is even possible) to alter typos in placeable descriptions. For this, I still use a complete module update, but there *are* ways to allow a player to carry on playing without having to start again even with this! It involves using Campaign Variables, and an example of this in operation can be seen in my module, Soul Shaker

So, there you have it: A reasonably simple way to keep modules "patched" and allowing players not to have to start again, which I know from experience, is sometimes too painful to bear.

Peter Thomas
Designer



Posted: Monday, 16 July 2007 10:07AM

This was basically the method used in the Infinite Dungeons premium module to allow players to add their own items, creatures, etc. The module read in 2das to which the players added their own objects and assimilated it into the appropriate systems.

Brian Chung

Tree Artist



Posted: Monday, 16 July 2007 12:07PM

Yes, it's the way resources work in NWN from what I understand. The system reads in order of module -> override -> bottom most HAK -> topmost HAK. So whatever is in the topmost HAK has the highest priority and overrides everything else.

Of course, HAKs, ERFs and MODs are just resource files to NWN (in fact, you can open up an ERF or MOD in the HAK editor). So anything can be into the HAK: models, textures, scripts, conversations, areas...

Brian Chung

Tree Artist



Posted: Monday, 16 July 2007 04:50PM

PWs can also make use of this by having a top most server_update.hak and using some free tools like rsync or wget, and the erf.exe tool, let people update the hak prior to joining the server - only download the changed files, package it up automatically then run "nwmain.exe +connect (ip)".

Obviously you'd have to move the updated stuff from the update hak to the main ones on, say, a monthly basis and refresh them - and if they already have the rsync/wget tools, it can download them for you automatically.

19.3 Alternate method for deleting Blueprints from your module

Deleting blueprints is quick and easy if you know the resref of it. Open the module in the toolset. Then browse your directories(not in the toolset now) and go into the folder /NWN/modules/temp0/ and find the filename that matches the resref, followed by the appropriate resources type extension (utc = creature, uti = item, etc.) and delete the file there. This will take only as long as it takes you to find the file. Then save the module in the toolset, and the resource is removed from the module.

The only thing lacking in this method of deletion is that there's no possibility to remove all references to the deleted resource.

19.4 Alternate method for deleting scripts from your module

Go to File -> Export -> Add Resources and select scripts from the dropdown. Highlight all the scripts you want to delete and press Delete. It's safer this way since you don't get to delete other module resources accidentally. You'll also get the references prompt thing, so you can remove any association at the same time.

CHAPTER 20 Common Acronyms

20.1.1 General Online Communications

Afaik - As far as I know

Afk - Away from keyboard

Asap - As soon as possible

Brb - Be right back

Bbiab - Be back in a bit

Bbl - Be back later

Btw - By the way

Fyi - For your information

IE - Internet Explorer/Infinity Engine/Infinity Explorer

IIRC - If I recall correctly

Imho - In my humble/honest opinion

ISP - Internet service provider

LARP - Live Action Role Playing

Lmao - Laughing my (expletive Deleted) off

Lol - Laughing out loud

Ltns - Long time no see

mIRC - Standalone IRC chat program used to reach SP chat rooms (setup instructions)

Nm - Never mind

Np - No problem

NS - Netscape

Omg - Oh my god

Otoh - On the other hand

Rotfl - Rolling on the floor laughing

Comment: Also may need some description or interpretation. That is, I noticed that PM is "Pale Master", but was not also included in the list so far as "Private Message". So a word about the context in which a term is used will indicate which of several meanings is being used. Also, seemingly minor differences can have significantly different meanings: PrC is Prestige class, but PRC is Player Resource Consortium.

Rotflmao - Rolling on the floor laughing my (expletive Deleted) off

Rtfm - Read the (expletive Deleted) manual

RTFM - Read the (expletive Deleted) Manual!

Rtffp - Read the (expletive Deleted) front page

Tmi - Too much information

Ttyl - Talk to you later

Ty - Thank You

Wb - Welcome back

what - What the (expletive Deleted)

Yw - You're Welcome

eg - Evil grin

g - Grin

20.1.2 Gaming Specific Terms

(A)D&D - (Advanced) Dungeons & Dragons, may be a reference to the first or second editions of the PnP game NWN and NWN2 are based on

(C)RPG - (Computer) Role Playing Game

3e - Third Edition Dungeons & Dragons rules

AA: Arcane Archer (a PrC)

BG - Baldur's Gate

CEP: Community Expansion Pack: a compilation of popular hak paks cleaned up and compiled by members of the NWN community, widely accepted by players.

CoT: Champion of Torm (a PrC)

D&D: Dungeons and Dragons

D&DO - Dungeons & Dragons Online

DA - Dragon Age

DD: Dwarven Defender (a PrC)

DM: Dungeon Master (someone who acts as referee and has administrative powers in D&D)

DSotSC - Dark Side of the Sword Coast (Baldur's Gate mod)

FR:DS - Forgotten Realms: Demon Stone

HCR – Hard Core Rules

HotU - Hordes of the Underdark (NWN expansion #2)

HoW - Heart of Winter (Icewind Dale expansion)

HS: Harper Scout (a PrC)

IWD - Icewind Dale, usually refers to the computer game of that name but is also a location in the Forgotten Realms campaign setting

MMORPG - Massively Multiplayer Online Role Playing Game

MP: Multi Player / Multi Player Module

NPC: Non-Player Character

NWN - Neverwinter Nights

NWN2 – Neverwinter Nights 2

OC: Official Campaign (the story ("campaign") that ships with NWN)

P&P - Pen & Paper (old school gaming from that ancient time before computers)

PC: Personal Computer (usually non-Mac), or Player Character, depending on context

PM: Pale Master (a PrC); also private message

PnP: Pen and Paper (this refers to the way D&D is classically played, with pen, paper, dice and most importantly, imagination)

PoR: RoMD - Pool of Radiance: Ruins of Myth Drannor

PrC: Prestige Class special character classes with advanced requirements

PS:T - Planescape: Torment

PvE: Player versus Environment (see PvM)

PvM: Player versus Monster (a style of play that emphasizes players fighting NPCs, usually in cooperation with other players)

PvP: Player versus Player (a style of play that emphasizes players fighting other players)

PW: Persistent World

RDD: Red Dragon Disciple (a PrC)

RPG: Role-Playing Game (should be self-explanatory...)

SD: Shadowdancer (a PrC)

SoA - Shadows of Amn (Baldur's Gate 2)

SoU - Shadows of Undrentide (NWN expansion #1)

SP: Single Player / Single Player Module

SPM: Single Player Module

ToB - Throne of Bhaal (Baldur's Gate 2 expansion)

ToEE - (The) Temple of Elemental Evil

TotLM - Trials of the Luremaster (Heart of Winter free expansion)

TotSC - Tales of the Sword Coast (Baldur's Gate expansion)

TTV - Table Top Variant Rules

WM: Weapon Master (a PrC)

WotC: Wizards of the Coast, a subsidiary of Hasbro. They publish the D&D PnP games, which is what NWN and NWN2 is based on.

CHAPTER 21 Resources

[Neverwinter Nights Vault](#) a resource for all manner of things related to Neverwinter Nights.

[GestaltCamera Scripting System v2.0](#) by John "Gestalt" Bye. This is a very popular scripting resource for creating cutscenes.

[The DMFI](#) The DM-Friendly Initiative (DMFI) is devoted to supporting DM-led gaming in Neverwinter Nights (NWN) and its successor, NWN2. [On the web.](#)

[DMFI Wand & Widget Package](#) The Wand & Widget Package is a compilation of tools created to expand the potential of the Neverwinter Nights DM client. DMs are given extensive new powers to manipulate the game, such as "throwing" their voice, playing a sound effect, or making skill checks. The tools also enhance the player experience by providing access to additional "emotes" and the ability to speak racial languages. The DMFI Wand & Widget Package is maintained by Hahnsoo, who also originally compiled the package and is a major contributor.

[Camtasia by Techsmith](#) a utility for creating AVI and other movie type files.

[StillListener's Corner](#) a free software site.

[LiveJournal](#) is a simple-to-use communication tool that lets you express yourself and connect with friends online.

[MySpace](#) Join for free and blog and do other social things.

[The Guide to Building – The Toolset Manual](#) looks at the toolset and the many options contained therein. The Toolset Manual is a reference manual in which every object type, property, tool and functionality offered by the Toolset is described in detail, along with many tips to help builders in the creation process.

[Lilac Soul's NWN Script Generator V2.2](#) a script writing tool widely used by the community.

[NWN Lexicon](#) is a complete scripting reference for BioWare's Neverwinter Nights role playing game. It is available as an up to date, online version at www.nwnlexicon.com, a downloadable Windows Compiled Help (*.chm) file, and a pure downloadable [HTML version](#) The current version is November 2004.

[The Hermit's Chalice](#) by Firestarter. A mod-building tutorial

[Custom Content Guide v3.0](#) it has everything you need to get started. The tutorial includes chapters on: - Tools, Weapons, Portraits, Voice Sets, Items Without Models, Spells, Custom Armour, Creatures, Custom Races, Visual Effects and a host of links to the more common tutorials on placeables, etc.). At the back is a summary drawing together the URL's for all of the tools you need to create custom content and then some.

[Custom Content Guide V3-1 Conversion on NWN wiki](#)

[NWN Wiki](#) fan website

[Sticky: HotU / 1.6x: New Scripting/Toolset Functionality explained](#) a BioWare forum thread that explains some of the new content introduced with the Hordes of the Undedark expansion.

[Module Builder's Henchman Kit v2.3a](#) Provides an intensive basis for the module builder to easily incorporate henchmen into their module.

[Guide to making HotU/SoU henchmen](#) This is a guide to making SoU and HotU henchmen.

[Programming Special Items Using Tag-Based Scripts](#)

[PW Helper](#) persistent world helper scripts.

[The Hitchhiker's Guide to Color Tokens](#) This short write-up is an attempt to explain everything color token. Upon reading this, you should be able to create, with next to zero effort, text strings of **any color** anywhere (almost)

There are a huge number of other guides and tutorials available on the Bioware forums and on the Vault

[Search Forums](#)

[Neverwinter Vault](#)

CHAPTER 22 CREDITS

Editors: qworty, Ghostdreamer,

Main Writer: Akkei

Content Contributors, Proof-readers, and Special friends. In alphabetical order:

AlexTiefing, Aliktren Stoutbow, Axe Murderer, Dallo, ffbj, gaoneng, Jasperre, Kosmous, Lance Botelle Mermut, rdjparadis, shadguy , Skymir, Sunjammer, Sunjammer, Thales Darkshine, Timendainum,

The members of The Builders Project Guild Source:

Most comments and recommendations for object script events were taken from the Lexicon (<http://www.nwnlexicon.com/>), the NWN scripting bible.

The "How to..." sections of this manual were mainly inspired by questions asked in the Builders Project help desk forum.

The "Designing tips" sections of this manual were mainly inspired by comments left on various custom made modules' page at NWVault (<http://nwwvault.ign.com/index2.shtml>) and in the play testing division forum of he Builders Project guild.